


Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Facultad de Informática

TRABAJO FIN DE GRADO

# Desarrollo de un prototipo usando como dispositivo de interacción Leap Motion

Autor: Karim Laazizi Ruiz

Tutor: Ángel Lucas González Martínez

MADRID, JUNIO DE 2014

# AGRADECIMIENTOS

*Durante el desarrollo de este trabajo de fin de carrera son varias las personas a las que les debo mi más sincero y profundo agradecimiento.*

*A mi familia y mis queridos padres que sin su ejemplo, su esfuerzo y sacrificio no hubiera podido alcanzar este primer objetivo. Siempre han estado a mi lado apoyándome, dándome ánimo y aconsejándome a lo largo de mis estudios y especialmente en los momentos más difíciles de la carrera.*

*A mi tutor que con su inestimable ayuda, dedicación, disponibilidad, sabiduría, guía y profesionalismo ha permitido que este trabajo tome forma, madure y sea mejor con sus correcciones y sugerencias.*

*A mis compañeros de carrera y amigos por su colaboración, su amistad y su apoyo incondicional durante este largo recorrido académico.*

*A todos, ¡muchísimas gracias!*

# Índice

# Índice de figuras

# 1. RESUMEN

## 1.1. Español.

Leap Motion [1] es un pequeño dispositivo que se coloca frente al monitor, conectado mediante un cable USB al ordenador, capaz de capturar los movimientos de nuestras manos y dedos con alta precisión, además de algunos objetos como pinceles o bolígrafos.

En la siguiente figura se puede ver el dispositivo Leap Motion.

**Figura 1: El dispositivo Leap Motion [2].**

Este aparato de interacción cuenta con 2 cámaras monocromáticas y tres Leds infrarrojos [3], permitiéndonos interactuar con el ordenador de forma intuitiva y simple mediante el uso de nuestras manos.

La página oficial nos ofrece las herramientas necesarias para crear aplicaciones que usen dicho dispositivo, haciendo más fácil su desarrollo y abriendo infinitas posibilidades.

**Figura 2: Visualización de las manos en 3 dimensiones [4].**

El objetivo principal de este trabajo es evaluar las capacidades de este dispositivo y crear un prototipo que sea capaz de grabar y reconocer gestos para que pueda ser fácilmente integrado a cualquier aplicación.

Para ello, el prototipo consta de 2 funciones principales:

- Grabar un movimiento: en el que recojo los datos que nos ofrece el Leap Motion, los proceso y los guardo en un formato específico.
- Reconocer un gesto: en el que comparo en cada momento el gesto que se está realizando con los gestos grabados mediante un algoritmo que detectara si son similares o no.

Este es un resumen básico del prototipo, sin embargo debemos tener en cuenta una serie de requisitos y parámetros para hacerlo más eficiente y personalizable dependiendo de las necesidades del usuario. Todos estos aspectos se explicaran en detalle en los siguientes apartados.

Una de las finalidades de este prototipo es su futura integración al proyecto que realice durante mi prácticum, que consiste en un juego educativo para niños con necesidades educativas especiales, en el que se usan diferentes dispositivos de interacción. Por ahora se pueden usar en dicho juego 3 dispositivos, siendo: la Kinect, la pantalla táctil y el teclado.

Este trabajo se realizó en el grupo de investigación CETTICO, situado en el bloque 1 de la Facultad de Informática (Ahora denominada Escuela Técnica Superior de Ingenieros Informáticos) de la Universidad Politécnica de Madrid, supervisado por el profesor Ángel Lucas González Martínez.

## 1.2. Inglés.

Leap Motion [1] is a small device we place in front of the display unit, connected to a USB cable to the computer. It is able to capture the motion of our hands and fingers with high accuracy, as well as some objects such as pens and brushes.

In the image below we can see the device Leap Motion.

**Figure 3: The Leap Motion device [2].**

This device includes two monochrome cameras and three infrared LEDs [3], allowing us to interact with the computer in a simple and intuitive way using our hands.

The official website provides us the required tools to create applications that use this device, making it easier to develop and opening endless possibilities.

**Figure 4: Visualization of the hands in 3 dimensions [4].**

This project's main goal is to evaluate the proficiency of the device, and create a prototype that is able to record and recognize gestures in order for it to be easily integrated into any application.

For that, the prototype has 2 main functions:

- Recording a motion: in which I collect the data offered by the Leap Motion, process it and keep it in a specific format.
- Recognizing a gesture: in which I compare each time the gesture being made with the gestures recorded using an algorithm to detect whether they are similar or not.

This is a basic summary of the prototype, but we need to take into consideration a number of requirements and parameters to make it more efficient and customizable depending on the user's needs. All these aspects are explained in detail in the following sections.

One of these purposes is its integration to the project that I made during my practicum the previous semester, which consists of an educational game for children with special needs, where different interaction devices are being used. So far, three devices can be used in this game: the Kinect, the touchscreen and the keyboard.

This work was done in the CETTICO research group, located in the block 1 of “La Facultad de Informática” (now called “Escuela Técnica Superior de Ingenieros Informáticos”) at the “Universidad Politécnica de Madrid”, supervised by Professor Ángel Lucas González Martínez.

# 2. INTRODUCCIÓN Y OBJETIVOS

## 2.1. Antecedentes del proyecto.

Hoy en día existen diversos dispositivos de interacción que facilitan la comunicación entre personas y ordenadores. Dicha interacción fue evolucionando con el tiempo para mejorar la experiencia del usuario.

Durante el primer semestre del 2013-2014, estuve desarrollando en el laboratorio CETTICO un juego educativo para niños con necesidades especiales, en el que se podían usar diferentes dispositivos de interacción. El niño podría elegir el dispositivo con el que se sentiría más cómodo.

En la siguiente imagen se puede ver el menú principal del juego desarrollado durante el primer semestre.


**Figura 5: El juego desarrollado durante el prácticum**

Por ahora se puede jugar con 3 dispositivos, siendo:

- La Kinect, poniéndose de pie delante de la cámara y haciendo gestos con los brazos.
- El teclado, usando las flechas y la tecla “espacio”.
- La pantalla táctil, usando el dedo para interactuar con los elementos del juego.

La idea fue añadir el nuevo dispositivo que salió al mercado en 2013, llamado Leap Motion, con el que interactuamos con el ordenador mediante gestos en el aire usando nuestras manos y dedos.

De este modo, podríamos evaluar las características y capacidades de este dispositivo innovador, así como su usabilidad.

## 2.2. Descripción del proyecto.

El proyecto se centra en la creación de un prototipo que permita grabar gestos que luego puedan ser reconocidos mediante el Leap Motion. Esto nos permitirá poder definir movimientos con las manos, e integrarlo a cualquier aplicación.


**Figura 6: Probando Google Earth mediante el Leap Motion.**

Un ejemplo básico seria: Si en alguna aplicación existe un menú en el que se puede navegar mediante las flechas del teclado, se grabarían movimientos con la mano haciendo gestos hacia la derecha e izquierda que se interpretarían de la misma manera.

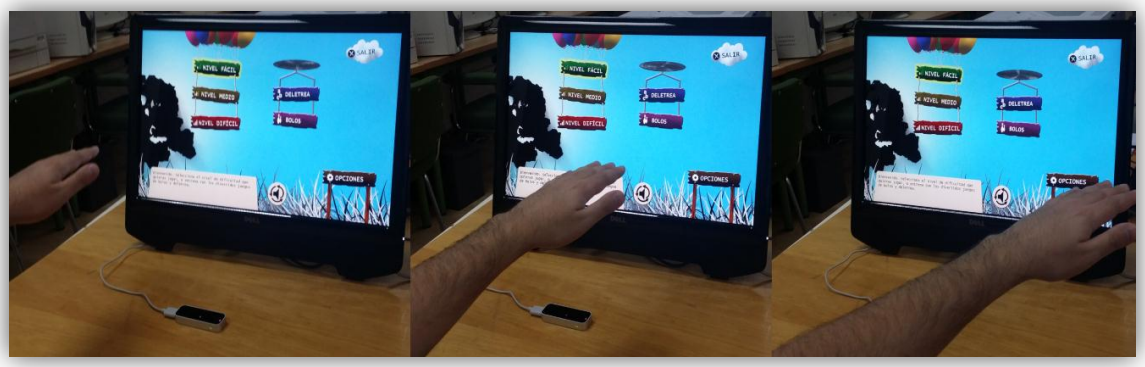


**Figura 7: Realizando el gesto hacia la derecha.**

Si una persona tiene una aplicación para pc, y le quiere añadir la interacción mediante el Leap Motion, solo tendrá que crear los gestos necesarios para controlar su aplicación e integrarlos. Por ejemplo, si la aplicación se controla mediante 6 teclas, entonces se tendrán que crear 6 gestos. Al hacer un gesto, se ejecutara la acción de su tecla

respectiva. Por lo tanto la incorporación del Leap Motion a cualquier aplicación sería muy simple.

Además, al crear un gesto, el usuario podría definir una serie de parámetros, como la velocidad, los ejes en los que se realiza el movimiento o el rango de similitud del gesto.

El proyecto se hizo para que fuese lo más personalizable posible, ya que un usuario podría querer un mismo gesto pero que dependiendo de la velocidad haga diferentes acciones, o que al hacer un movimiento, no tener en cuenta la altura o profundidad de la mano.

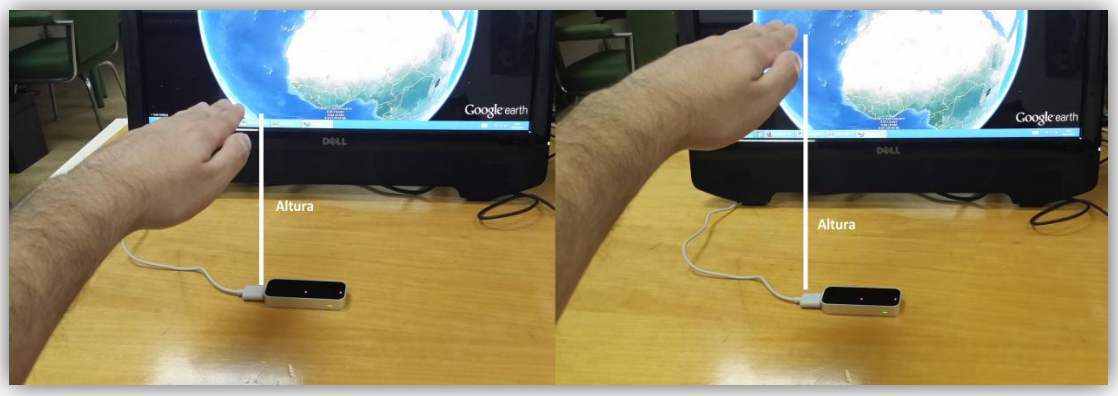



**Figura 8: El dispositivo Leap Motion detectando el movimiento con diferente altura.**

La aplicación del usuario almacenará los datos de los movimientos que usa, para que al realizar un movimiento compare con los gestos almacenados. Si se reconoce el gesto, se lanza un evento que captura la aplicación, y dependiendo del gesto, se realiza una acción u otra. En la siguiente figura se explica el procedimiento que se realiza:

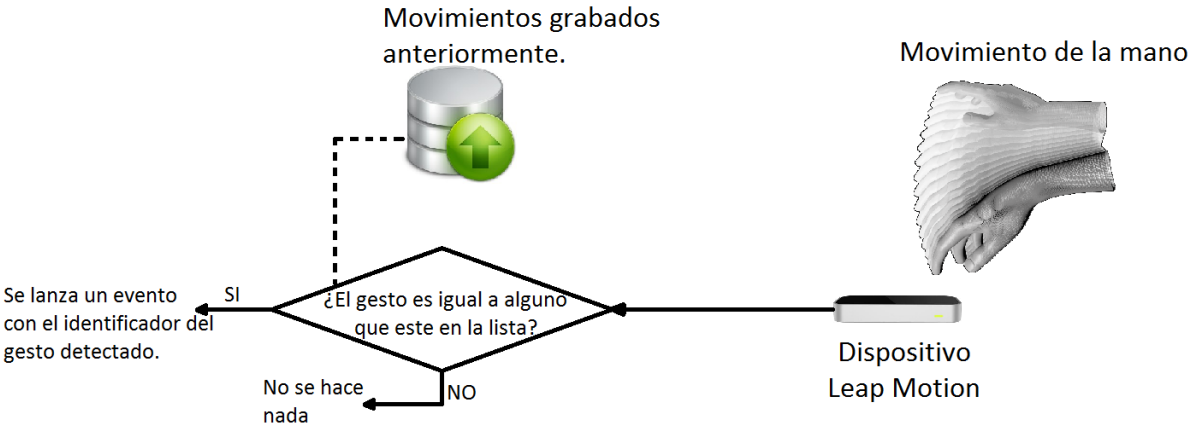



**Figura 9: Esquema del funcionamiento para lanzar eventos.**

## 2.3. Objetivos y problemas encontrados.

Una de las dificultades del proyecto fue la escaza información que existe del dispositivo dado que su salida al mercado sigue siendo reciente. Además, implemente el prototipo con un lenguaje de programación (C#) que nunca había usado.

Para afrontar dichos retos, tuve que documentarme sobre la API (*Application Programming Interface*) del Leap Motion y analizar las posibilidades que ofrece para crear y desarrollar programas usando dicho dispositivo.

Cuando aprendí a manejar la API, el objetivo principal fue diseñar e implementar un prototipo en el que se puedan grabar gestos. A cada gesto se le tendría que poder definir un nombre y un identificador. Además, al reproducir un gesto grabado se tiene que reconocer.

Los gestos se guardaran en un archivo XML, y el programa tendría que facilitar al usuario modificarlos, añadir nuevos movimientos, o eliminarlos. Además, al crear un gesto, este tendría que ser parametrizable.

## 2.4. Fases del desarrollo.

El proyecto se desarrolló a lo largo del segundo semestre del curso 2013-2014, y se realizaron aproximadamente 325 horas. Para ello, se creó un plan de trabajo y se dividió en 6 fases:

- **Fase 1: Documentación y lectura de la API.**

Esta primera fase consiste en documentarse sobre el Leap Motion para poder analizar su alcance y el uso que se le podría dar. Con ello se podrá definir los requisitos del prototipo y sus funcionalidades mínimas.

- **Fase 2: Preparación del entorno de trabajo.**

El objetivo de esta tarea es instalar y probar el entorno de desarrollo, para luego ejecutar ejemplos básicos. Esto nos es útil para comprobar que el compilador y el entorno de ejecución estén funcionando correctamente.

- **Fase 3: Estado de la cuestión.**

Se trata de la realización de un documento que consiste en la búsqueda y análisis de los prototipos, librerías y programas que usan Leap Motion y que puedan cumplir con alguno de nuestros requisitos. Esto nos servirá como referencia para saber lo que está hecho y poder reutilizarlo.

- **Fase 4: Definir los gestos y movimientos que se usaran en la aplicación.**

En esta tarea se definirán los gestos y movimientos que grabaremos para su uso en el juego educativo desarrollado durante mi prácticum.

- **Fase 5: Creación del prototipo.**
    - **Especificaciones.**

      Después de haber analizado el alcance del Leap Motion, se tendrán que definir los requisitos y las funcionalidades mínimas del prototipo, de las cuales son obligatorias: Poder grabar y reconocer un gesto.

    - **Diseño.**

      Una de las fases más importantes del desarrollo es la creación de la arquitectura del programa, en la cual haremos una estructura básica de nuestro prototipo.

    - **Implementación.**

      Después de saber cuáles serán las especificaciones y el diseño del prototipo, se tendrá que implementar el sistema cumpliendo con los requisitos definidos, siendo esta tarea la más larga.

    - **Pruebas.**

      Se harán pruebas a lo largo de la implementación para asegurarse de que el prototipo funciona correctamente.

- **Fase 6: Generación de la documentación.**

Esta fase se realizó a lo largo del semestre, y consiste en redactar varios documentos, tales como: los informes semanales, los informes de reunión, el plan de trabajo, el estado de la cuestión, la memoria de seguimiento, la memoria final y la presentación.

## 2.5. Estructura de la memoria final.

Esta memoria se ha estructurado de la siguiente forma para facilitar su lectura:

- **Resumen**:

En el que explico lo que es el Leap Motion e introduzco brevemente el proyecto para favorecer la comprensión del tema y facilitar su entendimiento.

- **Introducción y objetivos**:

Donde describo la razón por la que se realizó dicho proyecto, una descripción del trabajo realizado, los objetivos, las dificultades, las fases del desarrollo y la estructura del documento.

- **Estado del arte**:

En donde expongo el análisis de los prototipos, librerías y programas que usan Leap Motion y que puedan cumplir con alguno de nuestros requisitos, además del algoritmo que usaremos.

- **Desarrollo**:

En esta parte empiezo a hablar primero de los requisitos del prototipo, y hago un análisis del sistema en el cual describo las diferentes partes del proyecto. Después, explico en detalle cada parte de su funcionamiento.

- **Pruebas y resultados:**

En dicho apartado realizo diferentes pruebas con el prototipo y analizo los resultados obtenidos.

- **Conclusiones:**

En este apartado expongo las conclusiones del trabajo realizado.

- **Futuras líneas de desarrollo:**

Aquí describo las posibles futuras tareas que se podrían realizar para la mejora y continuidad del prototipo.

- **Bibliografía**:

En este apartado se citan las diferentes fuentes que he usado para la elaboración de este proyecto.

- **Anexo:**

Información adicional para el lector.

# 3. ESTADO DEL ARTE

## 3.1. Trabajos relacionados.

Este apartado consiste en una búsqueda y un análisis de los prototipos, librerías y programas que usan Leap Motion y que puedan cumplir con alguno de nuestros requisitos.

Estos programas podrán servirnos de referencia para la implementación de nuestro prototipo. En ellos podremos ver como se guardan los datos de los movimientos, y en algunos casos como se podrían comparar 2 gestos para saber si son similares o no.

Al ser un dispositivo nuevo, todavía existen pocos desarrolladores. La tienda online solo ofrece 206 aplicaciones [5], comparado con los más de un millón que ofrece Android [6]. De los diferentes programas estudiados, compararé los 3 que más se aproximan a nuestras necesidades, de los cuales: LeapTrainer.js, AirKey y JestPlay.

### 3.1.1. LeapTrainer.js [7].

Escrito en JavaScript, esta API es capaz de grabar diferentes gestos y movimientos para posteriormente ser reconocidos. Además, se puede integrar en cualquier aplicación web para reconocer los gestos grabados y asociarlos a una acción.

Para usarlo, se necesita el Leap Motion conectado a la máquina, y un navegador web abierto. El aparato envía los datos al navegador vía la API. Estos datos se analizan a continuación por el LeapTrainer.

Su funcionamiento es básico, al crear un nuevo gesto, se guardan los datos de dicho gesto en JSON [8]. Luego al hacer un movimiento, se comparan los datos de los gestos grabados con el movimiento mediante un algoritmo algebraico simple, y dependiendo del porcentaje de acierto, lo detecta o no.

El formato de los datos guardados sería parecido a:


```
{"name":"LEFT","pose":false,"data":[[{"x":0.4237907482090507,"y":-
0.02386414212986051,"z":0.12794371620034917,"stroke":1},
{"x":0.36582764056,"y":-0.00547780113,"z":0.06679630414,"stroke":1},
{"x":0.47232095409,"y":0.012939048550,"z":-0.0088868112,"stroke":1},
{"x":0.38230017519,"y":0.023041871250,"z":-0.0074585412,"stroke":1},
{"x":0.38014714589,"y":0.023046184250,"z":-0.008774542,"stroke":1}..
```


**Figura 10: Formato de los datos para el LeapTrainer.js.**

Como podemos ver, en el formato se guarda el nombre que le asignamos al gesto, el tipo, siendo o una postura (sin movimiento) o un gesto (con movimiento), y las coordenadas X Y Z en cada intervalo de tiempo.

En la siguiente figura se ve cómo se graba el movimiento ‘Derecha’, haciendo un movimiento con la mano de izquierda a derecha.


**Figura 11: Grabando el movimiento 'Derecha' con el LeapTrainer [9].**

En la siguiente figura pruebo el reconocimiento del gesto ‘Izquierda’ previamente grabado.

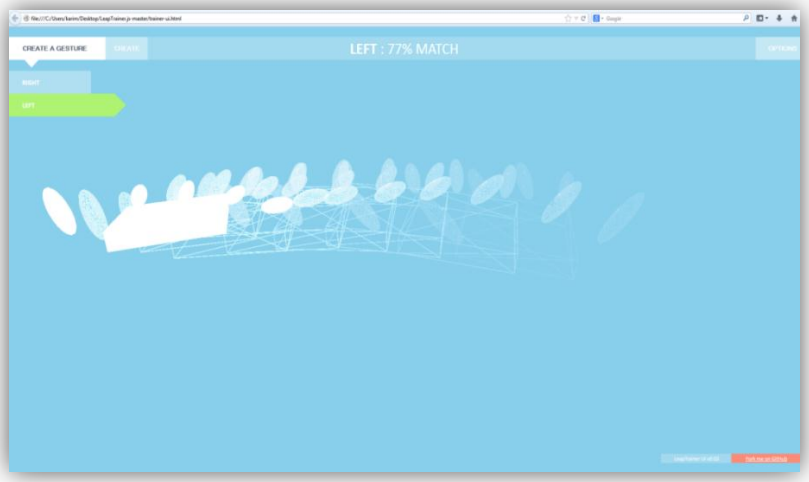



**Figura 12: Reconocimiento del gesto "izquierda" con el LeapTrainer.**

Aunque el programa este orientado a su uso en navegadores webs, al estar el código escrito en JavaScript, se podrían llamar a las diferentes funciones desde otras plataformas.

Sin embargo, al hacer varias pruebas, pude observar que el reconocedor no era muy preciso, ya que en pruebas básicas, algunas veces confundía el movimiento de la mano hacia la derecha con el movimiento hacia la izquierda. Esta API tampoco contempla el número de dedos en la mano ni la velocidad.

### 3.1.2. AirKey.js [10].

Esta herramienta no permite añadir más gestos de los que tiene, ya que solo se pueden usar los predefinidos. Se trata solo de un programa que se puede ejecutar en segundo plano para usar Windows mediante el Leap Motion.

Estos gestos están enlazados con los atajos del teclado (por ejemplo: Ctr+Alt+Sup) y pueden ser fácilmente configurable mediante un XML. Lo único que nos permite la herramienta es modificar los valores de dicho documento para configurar el atajo, por ejemplo:

```
<swipeUp>
<ctrl attr="true"></ctrl>
<alt attr="true"></alt>
<win attr="false"></win>
<shift attr="false"></shift>
<virtKeyCode attr="0"></virtKeyCode>
<keepPressed attr="false"></keepPressed>
</swipeUp>
```

**Figura 13: Ejemplo de configuración.**

En este caso, al hacer el movimiento "*SwipeUp*", un gesto con la mano llevándola de abajo hacia arriba, se ejecutarán las teclas CTRL+ALT.

Aparte de eso, también ofrece una interfaz básica para el uso del ratón mediante el dedo, pero no es de mucha utilidad ya que necesitamos grabar nuevos gestos para luego poder reconocerlos.

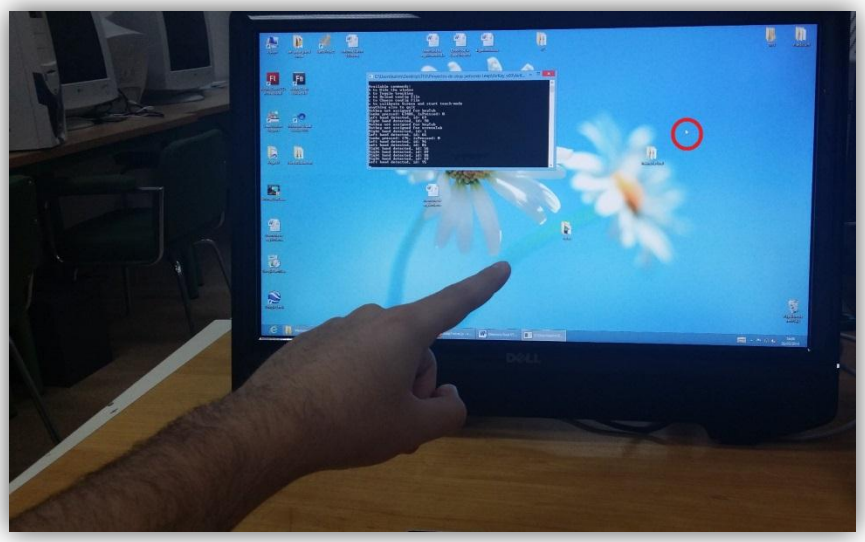


**Figura 14: Uso del ratón con el dedo mediante AirKey.**

### 3.1.3. JestPlay [11].

También en JavaScript, esta aplicación nos permite grabar nuestros gestos y movimientos guardando los datos que devuelve Leap Motion a nuestra máquina mediante BVH [12] o JSON.

El formato de los datos en JSON es parecido a la primera opción (LeapTrainer), pero con información adicional, como el número de dedos, la distancia entre cada dedo o los ángulos de cada miembro.

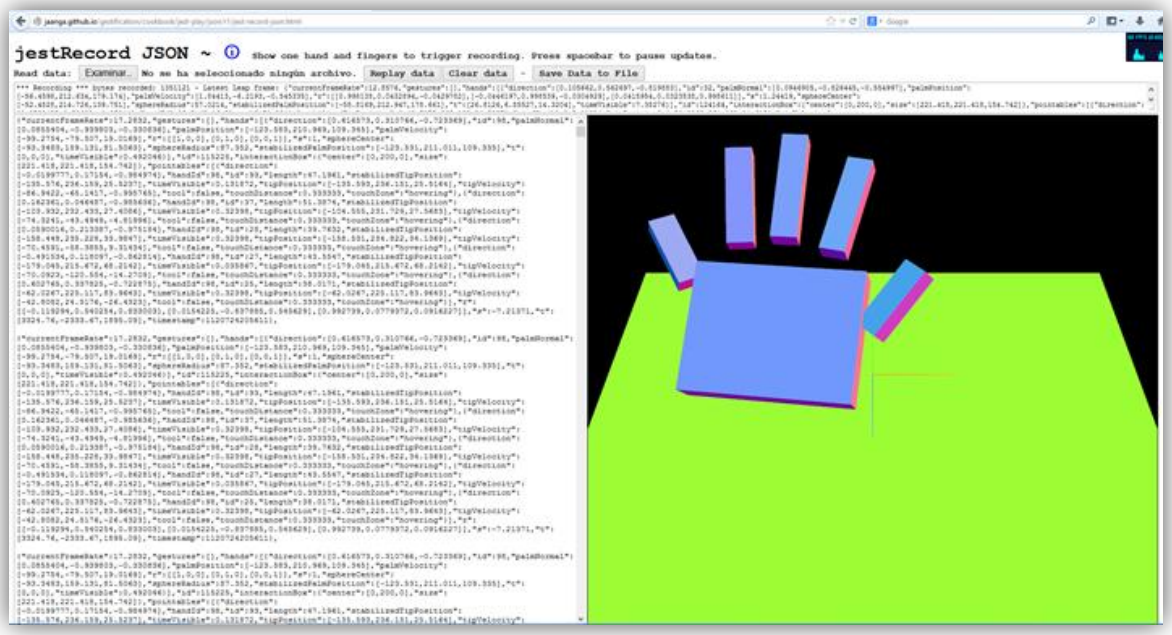



**Figura 15: Probando la reproducción en 3D de un movimiento.**

Al crear un movimiento, se obtiene un fichero con los datos, y estos se pueden introducir en el programa para que reproduzca el gesto visualmente. Por lo que al grabar la mano, la podemos reproducir en 3D.

Pero al igual que el LeapTrainer, al estar implementado en JavaScript, esta herramienta está orientada a priori solo para las aplicaciones web, aunque se podrían exportar y usar en diferentes plataformas.

Con esto lo que se podría aprovechar serían los datos que se obtienen al grabar un movimiento, por lo que tendríamos que crear un comparador para evaluar los gestos que se realizan con los que se tienen grabados y sacar un porcentaje de acierto para saber si son similares.

### 3.1.4. Desde cero.

Al detectar una mano, el Leap Motion nos ofrece una serie de información, de las cuales las más importantes son:

- El número de dedos.
- El número de manos.
- Las coordenadas X Y e Z de cada mano.
- Las coordenadas X Y e Z de cada dedo.
- La velocidad entre cada *frame*.
- Los ángulos de cada miembro.

Dichos datos se pueden ver en tiempo real mediante el programa aka.leapmotion [13]:

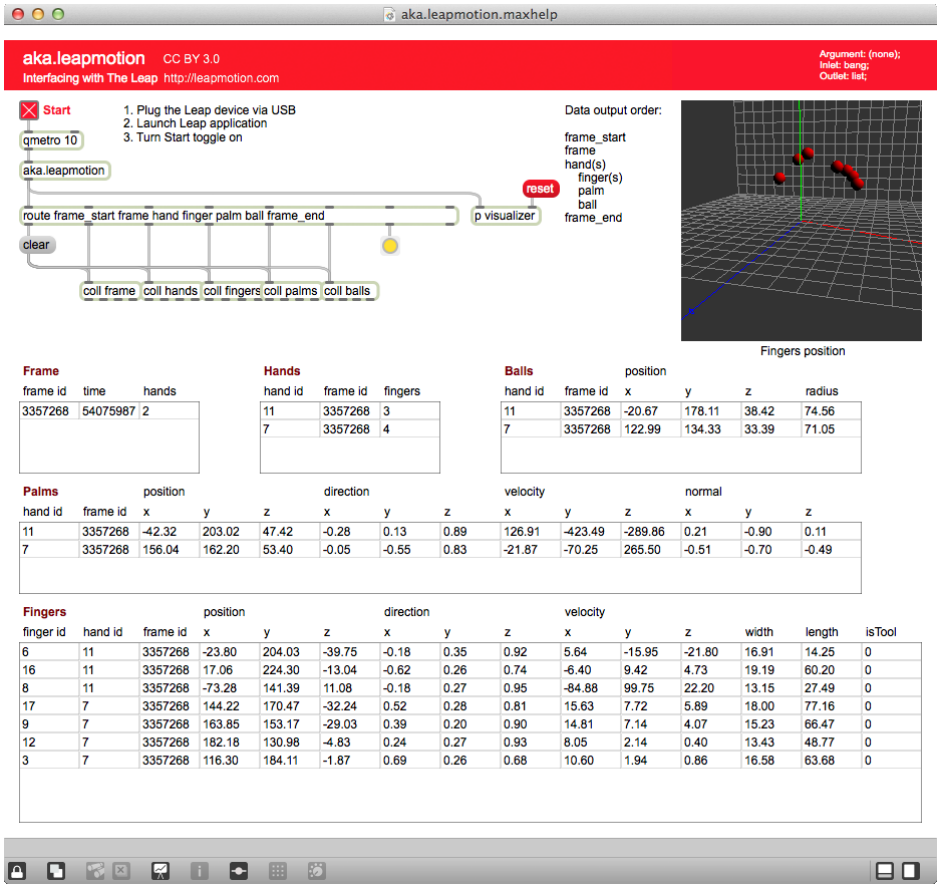



**Figura 16: Datos que se obtienen del Leap Motion con aka.leapmotion [14].**

Se pueden usar una variedad de lenguajes, como C++, C#, Java, JavaScript, Objective-C y Python. En el caso de empezar desde cero, usaríamos C# para que sea fácilmente compatible con el proyecto hecho durante mi prácticum.

Para la creación de un gesto, se podría recoger todo los datos de la mano y guardarlos en algún fichero, sea XML, CSV, JSON u otro.

Para la grabación de un gesto se guardarían los datos de la mano como la posición X Y Z en cada *frame*.

En la siguiente figura se puede ver el sistema de coordenadas del Leap Motion:

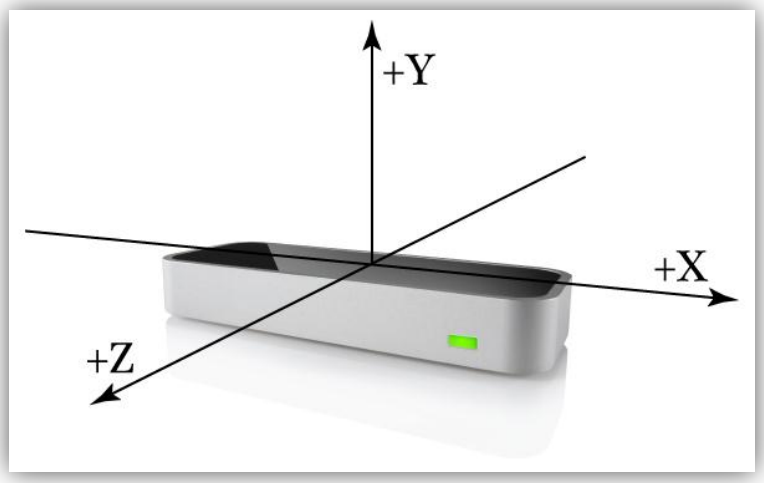



**Figura 17: Sistema de coordenadas del Leap Motion [15].**

Para el reconocedor, el algoritmo sería algo más complejo, pero en general, su funcionamiento básico sería: Detectar las manos que se encuentran en su campo de visión y empezar a recoger los datos del gesto. Estos datos se transformaran a un formato predeterminado, y se compararán con los gestos grabados. Si el porcentaje de acierto es alto con algún movimiento, se detecta.

Se podrían añadir opciones personalizadas, como por ejemplo: Tener en cuenta solo el eje X para los movimientos "izquierda" y "derecha", o solo el eje Y para el movimiento "abajo", todo eso dependiendo del movimiento que queramos grabar.

Los gestos dependen de varios factores, ya que por ejemplo, no sería lo mismo hacer un gesto rápido que lento, o mover la mano hacia la derecha con un número determinado de dedos o con el puño cerrado, por lo que se tendrían que tener en cuenta varios aspectos como el número de dedos o la velocidad para hacer el algoritmo más eficiente y preciso al intentar reconocer un gesto.

## 3.2. Algoritmo para el reconocimiento de gestos.

Para saber si un gesto es similar a otro, tendremos que comparar sus datos como la posición de la mano (o manos) en el eje X Y Z, el número de dedos y la velocidad del gesto.

Durante mi prácticum, en el juego implementado se usa el dispositivo Kinect, que detecta si un movimiento es similar a otro mediante el algoritmo DTW [16]. Dada la eficacia de este algoritmo para reconocer gestos, decidí usarlo en la implementación de mi proyecto.

### 3.2.1. El algoritmo DTW (Dynamic Time Warping).

El algoritmo DTW, en español llamado "Alineamiento Temporal Dinámico", sirve para medir la similitud entre dos secuencias temporales que pueden variar en el tiempo o la velocidad.

Originalmente se usó para el reconocimiento de voz, pero también se ha aplicado a otras áreas como las secuencias temporales de vídeo o datos gráficos. De hecho, cualquier dato que se puede convertir en una secuencia lineal puede ser analizado mediante el DTW.

En general, este algoritmo calcula una coincidencia óptima entre dos secuencias dadas. Estas secuencias no tienen por qué tener la misma longitud.

A = a1, a2, a3,..., ai,... an.
B = b1, b2, b3,..., bi,... bm

**Figura 18: Las secuencias A y B.**

Para entenderlo mejor usaremos un ejemplo sencillo [17]. Tenemos 2 secuencias A y B:


**Figura 19: Secuencia A.**

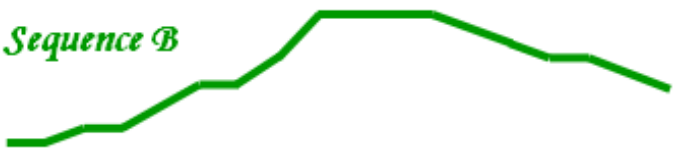



**Figura 20: Secuencia B.**

Estas 2 secuencias se dibujaran en los lados de una cuadrícula, uno en la parte superior, y otro en la parte izquierda. Dentro de cada celda se coloca la diferencia comparando los elementos correspondientes de las dos secuencias. Para encontrar la mejor coincidencia o alineación entre estas dos secuencias se tiene que encontrar un camino a través de la cuadrícula que minimiza la distancia total entre ellos. En la siguiente figura podemos ver el camino mínimo en forma de puntos rojos de la secuencia A y B.

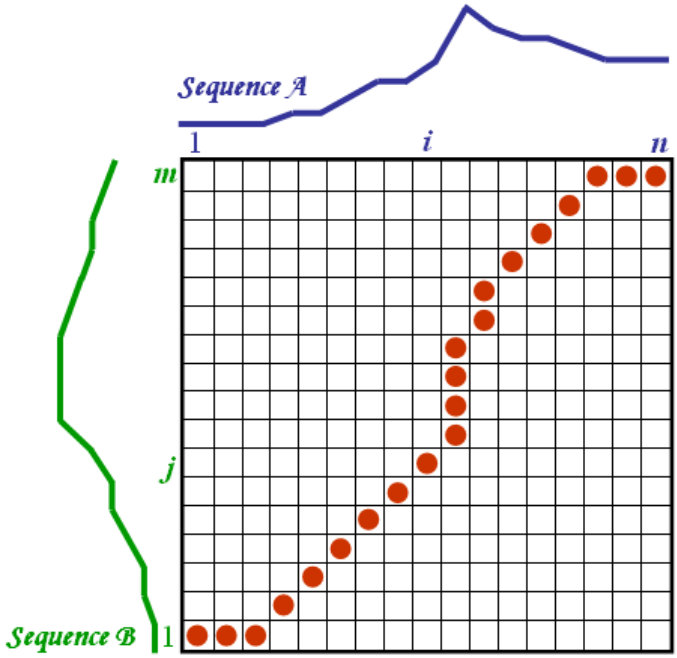



**Figura 21: Ejemplo al aplicar el algoritmo DTW [17].**

Por ejemplo, si tenemos la secuencia A=(2,3,5,7,10,6,3) y la secuencia B=(1,2,5,8,9,7,2), la figura seria:

| X | 2 | 3 | 5 | 7 | 10 | 6 | 3 |
|---|---|---|---|---|---|---|---|
| 2 | 0 | 1 | 3 | 5 | 8 | 4 | **1** |
| 7 | 5 | 4 | 2 | 0 | 3 | **1** | 4 |
| 9 | 7 | 6 | 4 | 2 | **1** | 3 | 6 |
| 8 | 6 | 5 | 3 | **1** | 2 | 2 | 5 |
| 5 | 3 | 2 | **0** | 2 | 5 | 1 | 2 |
| 2 | 0 | **1** | 3 | 5 | 8 | 4 | 1 |
| 1 | **1** | 2 | 4 | 6 | 9 | 5 | 2 |

**Figura 22: Ejemplo para el cálculo de las diferencias con el algoritmo DTW.**

El procedimiento para el cálculo de esta distancia en general consiste en encontrar todas las rutas posibles a través de dicha red y para cada una calcular la distancia total. En el ejemplo de la figura anterior, el camino optimo seria: 1+1+0+1+1+1+1+1 = 6.

Dado que un análisis en profundidad para crear una implementación de dicho algoritmo conllevaría mucho trabajo (aproximadamente el tiempo de un Trabajo de Fin de Grado), decidí usar una implementación ya hecha del algoritmo en C# con licencia MIT, que me permite usarlo y modificarlo sin restricciones.

La implementación elegida será el NDtw [18], que consta de 2 paquetes importantes:

- NDtw: El algoritmo.
- NDtw.Visualization.Wpf: Visualización grafica de los resultados.

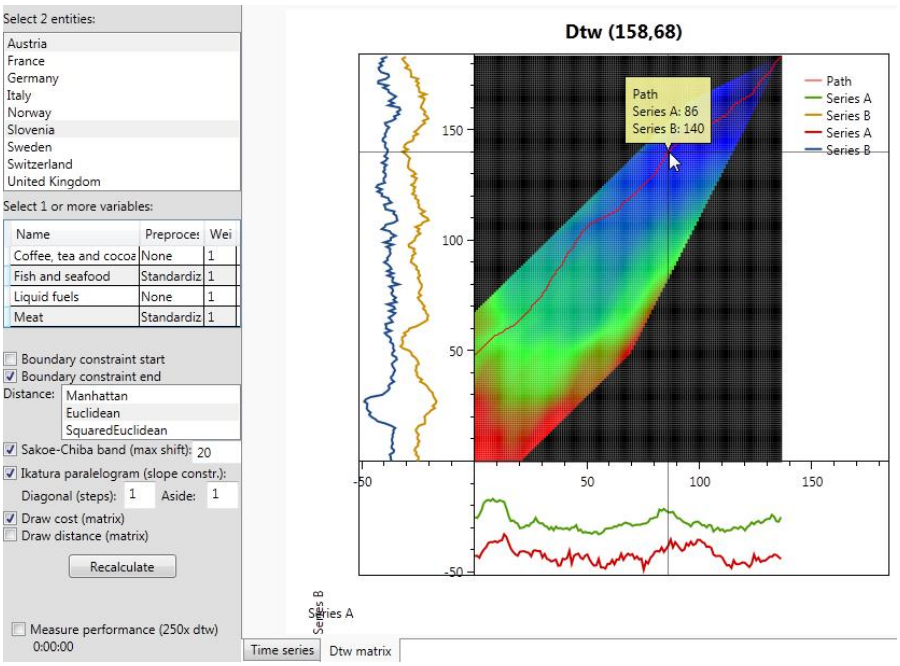



**Figura 23: Visualización de la comparación mediante NDtw.**

Esta implementación es fácil de usar, ya que solo se tiene que llamar al método y pasarle las 2 secuencias que queremos comparar. Además nos permite modificar diferentes parámetros al inicializarlo para hacerlo más eficiente.

## 3.3. Comparación de las diferentes vías de desarrollo.

### 3.3.1. Comparación de los prototipos analizados.

Para concluir, podemos observar en la siguiente tabla la comparación de las diferentes opciones que tendríamos.

| | LeapTrainer | AirKey | JestPlay | Desde cero |
|---|---|---|---|---|
| **Lenguaje** | JavaScript | - | JavaScript | C# (+1) |
| **Grabación** | SI (+1) | NO | SI (+1) | SI (+1) |
| **Reconocedor** | SI (+1) | Solo gestos predeterminados | NO | SI (+1) |
| **Usable** | Puede (+1) | NO | Puede (+1) | - |
| **Suma** | 3/4 | 0/4 | 2/4 | 3/4 |

**Figura 24: Tabla comparativa de las diferentes opciones.**

Como lo vimos, AirKey y LeapMotionP5 no permiten graban nuevos movimientos ni reconocerlos.

LeapTrainer y JestPlay están escritos en JavaScript, y permiten definir nuevos gestos, guardando los datos en un formato predefinido.

El JestPlay es más preciso, ya que al introducir los datos del movimiento, es capaz de reproducirlo en 3D, por lo que necesita los datos de cada dedo y sus ángulos. Sin embargo el LeapTrainer es más simple, y guarda pocos datos, siendo los necesarios para poder comparar gestos. La diferencia entre los dos es que el primero puede reconocer los gestos, y el segundo no.

### 3.3.2. Elección.

Para la implementación de mi proyecto, decidí empezarlo desde cero, creando una estructura para guardar los datos, ya que una conversión de JavaScript a C# sería muy costosa.

Para la comparación de gestos usare la implementación NDtw del algoritmo DTW.

# 4. DESARROLLO

## 4.1. Los requisitos.

Antes de empezar a implementar el prototipo, se establecieron una serie de requisitos para especificar su alcance.

En este apartado defino las funcionalidades mínimas que tendrá el prototipo, y los requisitos avanzados que se implementaran.

### 4.1.1. Requisitos básicos.

El prototipo tiene 2 funcionalidades mínimas:

- **Permitir definir gestos:**

El usuario tendrá que poder grabar un movimiento con su mano (o sus dos manos). Para ello tendré que recoger los datos del gesto, procesarlos y guardarlos en un formato específico.

Al crear un nuevo gesto, el usuario podrá asignarle un nombre y un identificador (que tendrá que ser único).

- **Reconocer un gesto:**

El prototipo tendrá que reconocer si el usuario hace un movimiento similar a otro ya grabado. Para ello, usare el programa NDtw [18] para comparar en cada instante el gesto que se está realizando con los gestos grabados anteriormente.

### 4.1.2. Requisitos avanzados.

Además de las funcionalidades mínimas, se establecieron otros requisitos avanzados para hacer el prototipo más eficiente y personalizable dependiendo de las necesidades del usuario.

- **Tener o no en cuenta la velocidad:**

Al crear un gesto, el usuario podrá definir si la velocidad importa o no. Si el gesto tiene un significado diferente al ser lento o rápido, el usuario podrá seleccionar en que eje se tiene en cuenta la velocidad.

Cuando se grabe un gesto teniendo en cuenta la velocidad, se usara un rango que podrá definir el usuario para reconocer los gestos. Por ejemplo: “El gesto llamado Izquierda será válido si la velocidad está comprendida entre los valores A y B milímetros por segundo”.

- **Tener en cuenta los ejes en los que participa el movimiento:**

Los sensores que dispone el Leap Motion nos permiten saber con precisión la posición de la mano (o manos) en los 3 ejes X Y Z.

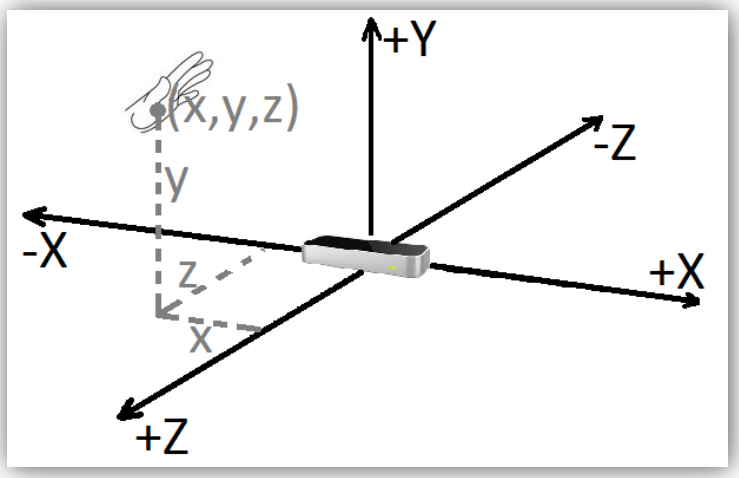



**Figura 25: Posición de la mano en los ejes X Y Z.**

El usuario podrá elegir los ejes en los que el movimiento tenga que variar, ya que por ejemplo:

Si se quiere grabar un movimiento diagonal de la mano desde la esquina inferior izquierda hasta la esquina superior derecha como lo muestra la siguiente figura:

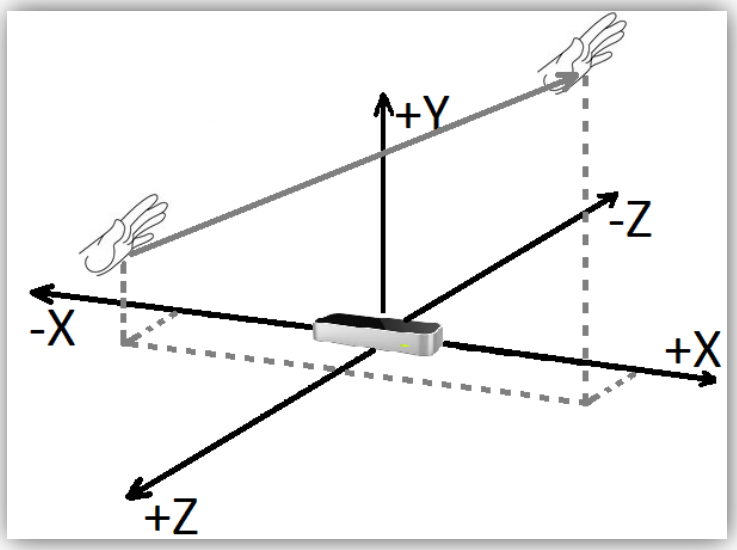



**Figura 26: Movimiento de la mano de abajo a arriba en diagonal.**

Pero no se quiere tener en cuenta la profundidad (El eje Z), o mejor dicho, que su valor siempre sea constante, los gestos que se reconocerán podrían ser los siguientes:

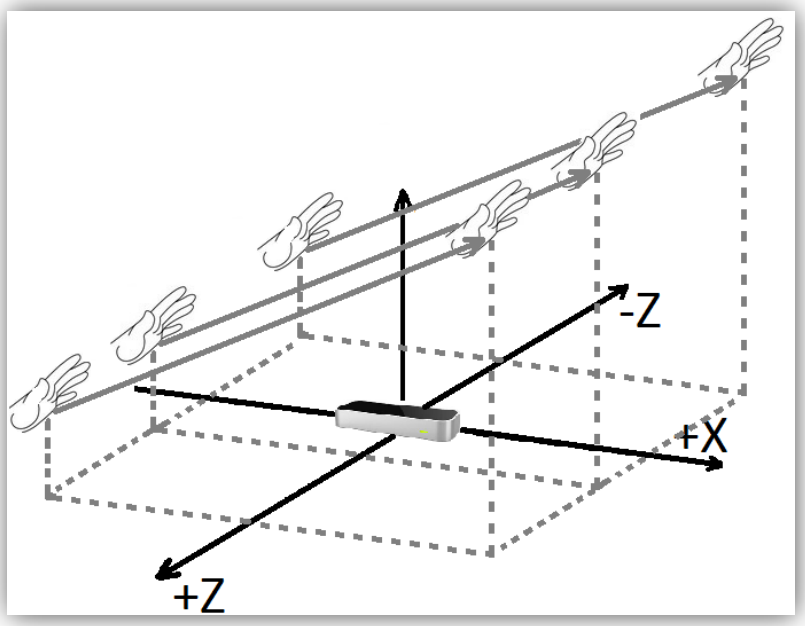



**Figura 27: Posibles movimientos que serán reconocidos.**

De esta manera, el movimiento se reconocerá desde cualquier profundidad mientras sea constante y se haga el movimiento diagonalmente.

Sin embargo este gesto no será reconocido:

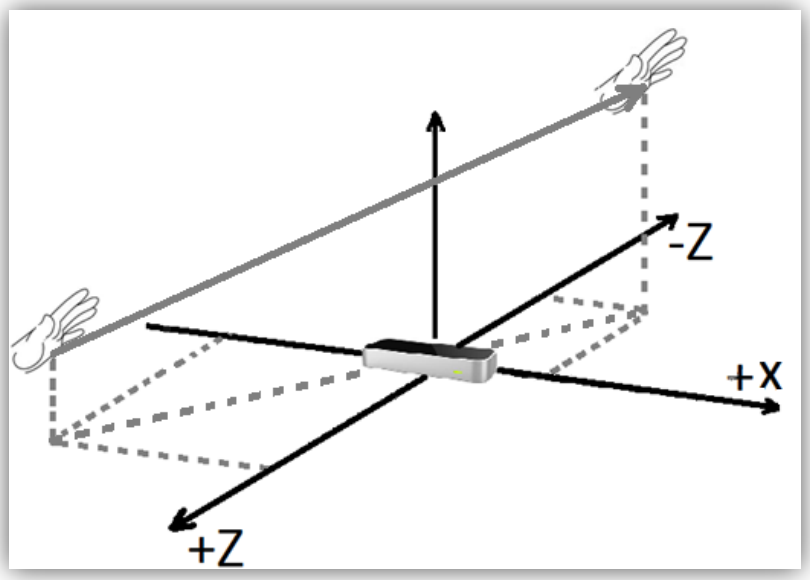



**Figura 28: Movimiento diagonal variando la profundidad.**

- **Implementar un gestor de gestos:**

Al grabar un gesto, el usuario podría añadir dicho gesto a un archivo con una lista de gestos ya grabados, o crear un nuevo archivo con dicho gesto.

El usuario podrá cargar el archivo al reconocedor para probar si se reconocen bien, y si queda insatisfecho, poder borrar algún gesto de la lista.

- **Interfaz gráfica:**

Crear una interfaz gráfica del prototipo para facilitar al usuario crear, personalizar, modificar y reconocer gestos.

- **Configuración del reconocedor:**

El usuario tendrá que ser capaz de modificar los valores por defecto que se usan, como por ejemplo: el coste de reconocimiento, el *frame* de inicio y fin para el cálculo de la velocidad media, el número de *frames* que tiene el gesto, entre otros.

## 4.2. Análisis del sistema.

En este apartado explicare las diferentes partes que interactúan en el prototipo y su funcionamiento para tener una idea general del sistema.

### 4.2.1. Los datos que se obtienen del Leap Motion.

Cuando se crea el escuchador del Leap Motion, este dispositivo nos devuelve una serie de imágenes instantáneas que capta en su campo de visión. Estas imágenes se llaman *frames*.

El dispositivo se sitúa centrado en frente del monitor, a 30 cm*. Su rango de visión es de una distancia de 40 cm* en el lado derecho e izquierdo, y de 35 cm* de frente y de detrás. El ángulo de visión entre el Leap Motion y la mesa es de 20 grados*. (*Distancias aproximadas que dependen de varios factores como la capacidad de cómputo o del entorno.)

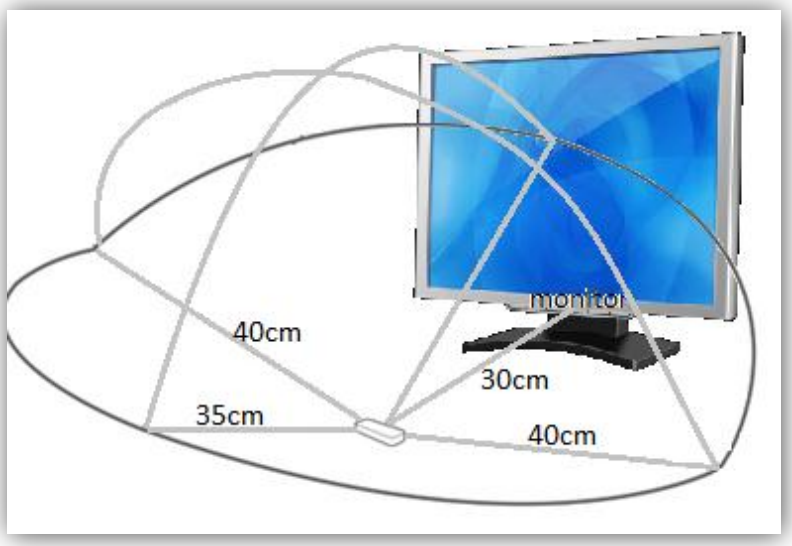



**Figura 29: Área de visión del Leap Motion.**

Cada *frame* que recibimos dispone de la información captada en el momento, de la cual se puede extraer:

- El número de manos.
- El número de dedos de cada mano.
- La posición en X Y Z de cada palma.
- La posición en X Y Z de cada dedo.
- Ángulo de la mano.
- Velocidad de la palma.

El número de *frames* por segundo que devuelve el dispositivo depende de los recursos de la máquina, el número de elementos en su campo de visión, los ajustes de seguimiento y otros factores.

Al hacer un movimiento con la mano obtendremos varios *frames* con sus respectivos datos:

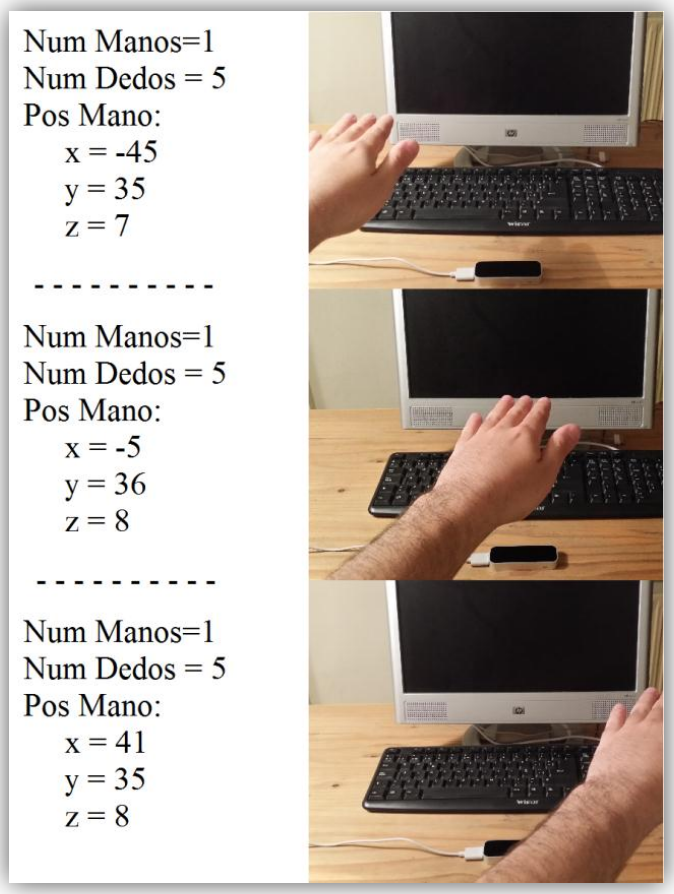



**Figura 30: Algunos datos de los *frames* que obtenemos al mover la mano.**

La información que obtenemos del Leap Motion se puede guardar en una estructura de datos, para posteriormente ser utilizada.

### 4.2.2. Componentes del proyecto.

En este proyecto existen 4 partes importantes que forman el prototipo, de los que hablare en detalle en el siguiente apartado.

- **La estructura de un gesto.**

Para empezar, tuve que crear la estructura que tendrá un gesto con los datos que necesitaremos para diferenciarlos. Esto nos servirá para guardar los movimientos y posteriormente poder compararlos.

Los gestos se guardaran en un archivo separado para facilitar su exportación. (Ver Anexo A.2)

- **El gestor de gestos.**

Para que el usuario no tenga que modificar los archivos que contienen los gestos, cree un gestor de gestos que permite añadir o eliminar un gesto del documento. Esto facilitaría su gestión.

- **La grabación de gestos.**

También hablare de los datos que guardo al recibir los *frames* del Leap Motion, y la forma en la que lo hago.

- **El reconocedor.**

Una de las partes más importante del proyecto, donde tengo que comparar en tiempo real el movimiento que se está realizando con los gestos grabados para determinar si son similares o no.

## 4.3. La estructura de un gesto.

### 4.3.1. Formato del documento.

Cuando se graban los gestos se tendrán que almacenar los datos en algún fichero de tipo JSON, BVH, XML o CSV con un formato predeterminado, para posteriormente poder ser reconocidos.

Para seguir el mismo patrón que se usó para el reconocedor de gestos para Kinect en el juego educativo, usaremos un archivo de tipo XML. Cada gesto tiene que tener los atributos:

- **Nombre:**

  Nombre del gesto grabado.

- **Identificador:**

  Identificador único, formado por 5 dígitos, que nos permitirá diferenciar los diferentes gestos.

- **Dispositivo:**

  Para saber de qué dispositivo son los datos. Por ahora puede tener el valor ‘Kinect’ o ‘Leap’. El valor por defecto es ‘Leap’.

- **Participa velocidad:**

  Por ahora solo se permite contemplar la velocidad en un eje determinado, por lo que los valores que puede tener son ‘X’, ‘Y’, ‘Z’ o ‘NO’. Si se quiere tener en cuenta la velocidad, se introduce el eje en el que se quiere considerar, sino se introduce ‘NO’.

- **Velocidad mínima:**

  La velocidad mínima que tiene que tener el gesto para ser reconocido. (Si se tiene en cuenta la velocidad)

- **Velocidad máxima:**

  La velocidad máxima que tiene que tener el gesto para ser reconocido. (Si se tiene en cuenta la velocidad)

- **Participa X:**

  De tipo booleano, define si el movimiento es variable en el eje X. Si el valor es ‘falso’, se reconocerá el gesto si el movimiento no varía en el eje X.

- **Participa Y:**

  Igual que el atributo anterior, solo que esta vez en el eje Y.

- **Participa Z:**

  Igual que el atributo anterior, solo que esta vez en el eje Z.

- **Lista de esqueletos:**

  Una lista de esqueletos que define el movimiento, siendo cada esqueleto la información de un instante. Si el gesto está constituido por 35 *frames*, el gesto tendrá 35 esqueletos.

  Cada esqueleto tiene una lista de *joints* (articulaciones):

  - **Lista de *joint*:**

    El *joint* define la articulación, que puede ser la mano derecha, izquierda, o algún dedo. Cada *joint* tiene un nombre y una posición.

    Para saber si el *joint* está involucrado en el gesto, tenemos la variable booleana 'involucrado'.

En la siguiente imagen podemos ver la estructura del documento:

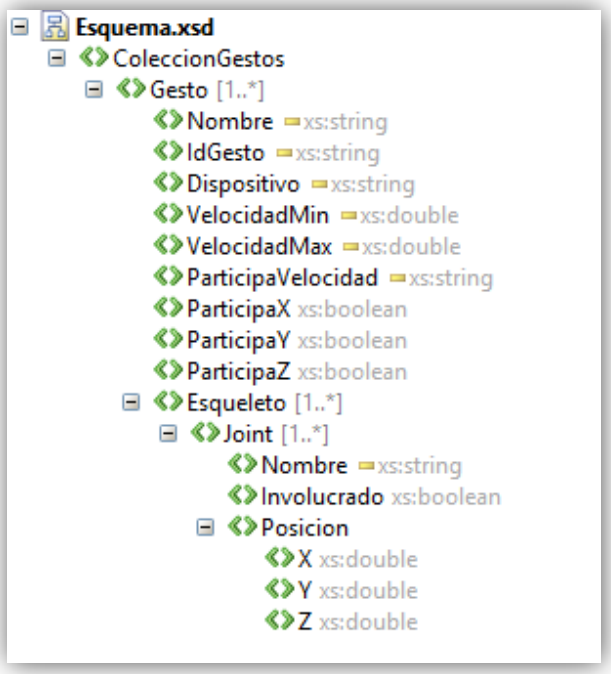



**Figura 31: Esquema general para guardar gestos en un XML.**

Se puede ver el esquema completo del archivo XML en el Anexo A.2.

### 4.3.2. El esqueleto.

La lista de esqueletos define el movimiento, y en cada esqueleto tenemos la información de las diferentes articulaciones que capta el Leap Motion en un determinado instante de tiempo.

Cada esqueleto contiene una lista de *joints.*

#### *4.3.2.1. El joint.*

El *joint* está formado por el nombre de la articulación, un atributo que define si está involucrado en el movimiento, y su posición en el eje X Y Z.

Por ahora se han definido doce articulaciones, siendo:

- La mano derecha.
- La mano izquierda
- Los 5 dedos de la mano derecha.
- Los 5 dedos de la mano izquierda.

Uno de los problemas encontrados ha sido que no se podrían diferenciar los diferentes dedos de la mano, por lo que se decidió no guardarlos de momento. Sin embargo, el 28 de mayo salió una nueva actualización del SDK que nos ofrecía dicha información. (Ver Futuras líneas de desarrollo.)

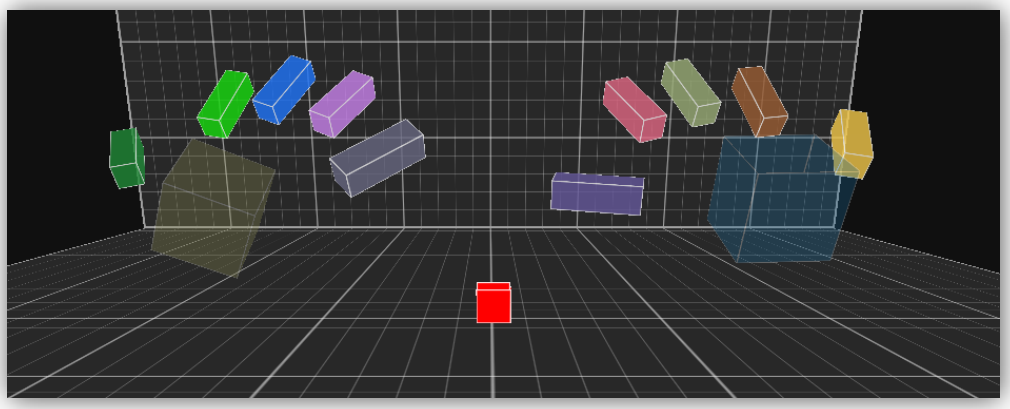


**Figura 32: Diferentes miembros que captura el Leap Motion.**

## 4.4. El gestor de gestos.

Para facilitar al usuario modificar un archivo con una lista de gestos, se decidió crear un gestor de gestos que ofrece diferentes funciones útiles para crear, modificar y eliminar un gesto de un documento.

### 4.4.1. Funciones del gestor.

#### *4.4.1.1. Carga de los gestos.*

Una de las funciones principales del gestor es cargar los gestos almacenados en algún XML para que luego puedan ser reconocidos.

Para empezar se verifica si el documento XML sigue las reglas del esquema y asegurarnos de que es válido. Luego procesamos sus datos para crear los objetos 'Gesto' para facilitar la comparación de gestos.

#### *4.4.1.2. Guardar un gesto.*

Otra de sus funciones principales es poder guardar un gesto en un documento XML. Existen 2 casos:

- Si es la primera vez que se quiere grabar un movimiento o si se quiere crear en un documento XML separado, se puede elegir la opción 'Nuevo documento'.

- Si se quiere añadir un gesto a una lista de gestos ya insertados en un XML, se puede elegir la opción 'Añadir a documento'.

#### *4.4.1.3. Eliminar un gesto.*

Se tiene la posibilidad de eliminar un gesto de la lista de gestos contenida en un documento XML mediante su identificador.

## 4.5. La grabación de gestos.

Para asignar una acción a un gesto el usuario tendrá primero que grabarla. Para ello se creó dicha función que graba los gestos y guarda los datos en un archivo. Dichos datos serán posteriormente usados para el reconocimiento.

### 4.5.1. Configuración de la grabación.

Cuando el usuario quiera grabar un nuevo gesto, podrá configurar una serie de parámetros:

#### *4.5.1.1. Lugar en el que se guarda el gesto:*

Se pueden crear diferentes documentos con listas de gestos, por lo que al crear un gesto, este último se puede añadir a un archivo que contiene ya uno o varios gestos, o crear un nuevo documento con el gesto grabado.

#### *4.5.1.2. Las propiedades del gesto:*

El gesto tiene una serie de propiedades que se le tienen que asignar, como el nombre y el identificador único para diferenciarlos.

También se tendrá la opción de tener en cuenta la velocidad en algún eje, pero para ello se establecerá una velocidad mínima y una velocidad máxima para su reconocimiento, que serán calculados mediante la velocidad media del gesto grabado. Si se vuelve a reproducir el movimiento con una velocidad entre el valor mínimo y máximo, se reconocerá.

Además, se podrán definir los ejes en los que el movimiento varia, como lo explicamos en el apartado 4.1.2.

## 4.5.2. Funcionamiento de la grabación.

Para la grabación de un movimiento se siguen una serie de pasos que son:

### *4.5.2.1. Introducción de los parámetros.*

Antes de lanzar la grabación, el usuario introduce una serie de parámetros, siendo:

- Nombre e identificador del gesto.
- Nombre del documento si se crea uno nuevo, o la ruta de un documento ya existente.
- Ejes en los que participa el movimiento.
- Si se tiene en cuenta la velocidad.

También podrá modificar algunos parámetros que están por defecto, como:

- *Frames* a grabar. Por defecto son 60, un tiempo aproximado de 2 segundos.
- *Frame* de inicio y de fin para el cálculo de la velocidad media.
- Valor para crear el rango de velocidad.

### *4.5.2.2. Lanzamiento de la grabación.*

Cuando el usuario lanza la grabación, se activa el escuchador del Leap Motion para la recepción de datos.

Para cada *frame* que llega:

1. Se guarda la posición de la mano (o manos) en el eje X Y Z.
2. Se guarda la velocidad del *frame*.
3. Si el número de *frame* es igual que el número de *frames* a grabar, pasamos al siguiente apartado. Sino volvemos al paso 1.

### *4.5.2.3. Calculo de la velocidad media.*

Para calcular la velocidad media, se suma la velocidad de cada *frame* y se divide por el número de *frames* grabados.

El problema principal es que al hacer un gesto, la aceleración al principio y al final del movimiento es muy baja, e influyen mucho sobre el resultado.

Para ello se decidió calcular la velocidad media a partir de un cierto *frame* hasta otro. Por defecto el *frame* inicial es igual a 15 y el *frame* final a 45.

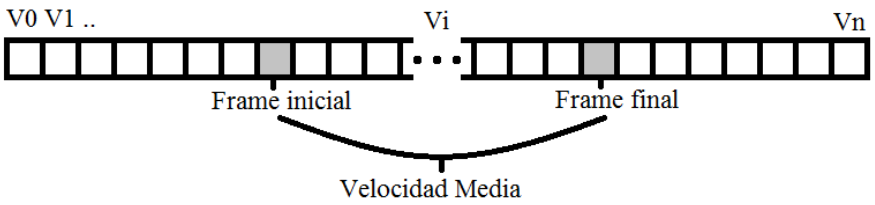



**Figura 33: *Frames* que se usan para calcular la velocidad media.**

#### 4.5.2.4. Guardar los datos en el documento.

Al ser muy complicado reproducir un gesto con la misma velocidad, usaremos un rango para que el gesto solo pueda ser reconocido si su velocidad media está entre la velocidad mínima y la velocidad máxima del movimiento.

Los valores por defecto para crear el rango son 50 y -50. Por lo que para calcularlo se suman dichos valores a la velocidad media del gesto grabado. Por lo tanto al grabar un gesto con una velocidad de 700 milímetros por segundo, el gesto será reconocible si su velocidad oscila entre 650 y 750 mm por segundo.

Teniendo todos los datos necesarios, se pasa a grabar el gesto mediante la función del gestor.

#### 4.5.2.5. Resumen.

En la siguiente imagen podemos ver un resumen del funcionamiento de la grabación:

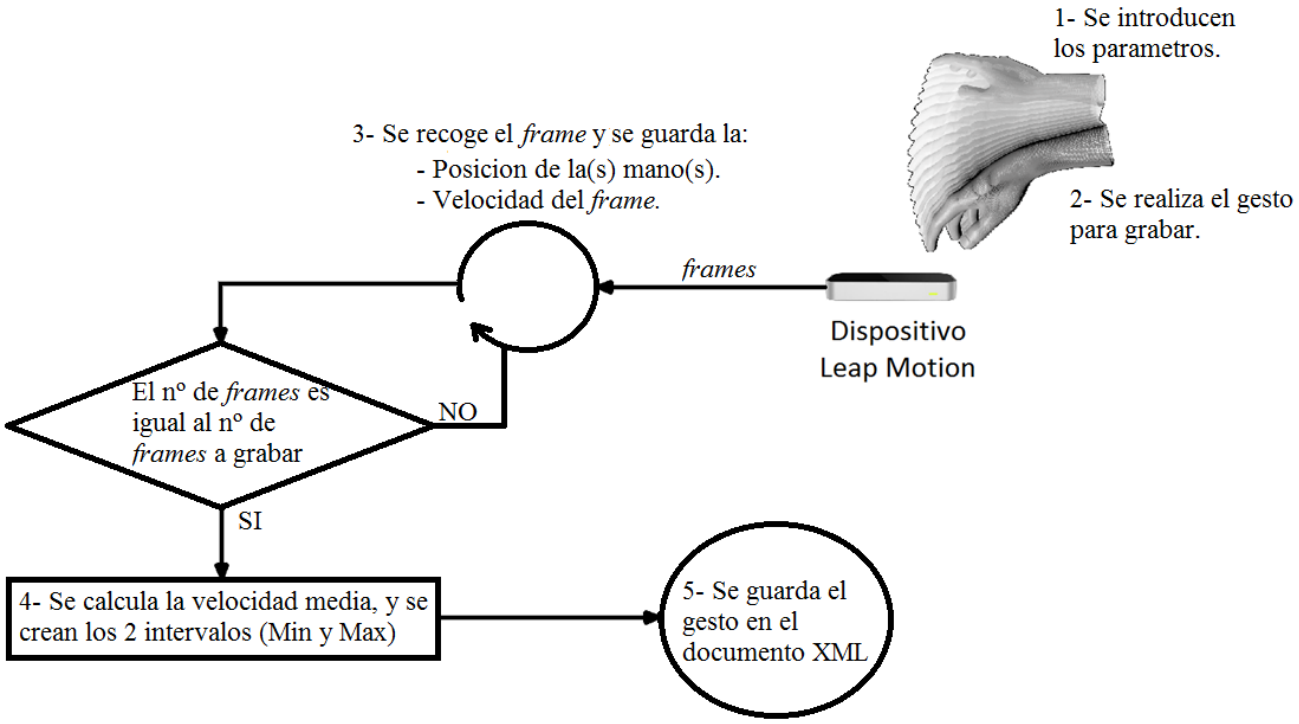



**Figura 34: Resumen del funcionamiento de la grabación.**

## 4.6. El reconocimiento.

Una de las partes más complejas del proyecto es el reconocedor, ya que se tiene que ir comparando el gesto que se está realizando en tiempo real con los diferentes movimientos cargados.

### 4.6.1. El Algoritmo.

Como he dicho anteriormente (Ver apartado 3.2), usaremos el algoritmo DTW, para comparar los diferentes gestos, y más precisamente la implementación NDtw [18]. Dicho algoritmo recibe como parámetros 2 secuencias de números que analizara para determinar el grado de similitud.

A continuación, explicare un ejemplo para entender cómo podríamos usar dicho algoritmo con los datos del Leap Motion:

Tenemos guardado el movimiento de una mano, lo que significa que tenemos una lista de posiciones en el eje X Y Z que lo definen. Dichos datos están guardados en el esqueleto.

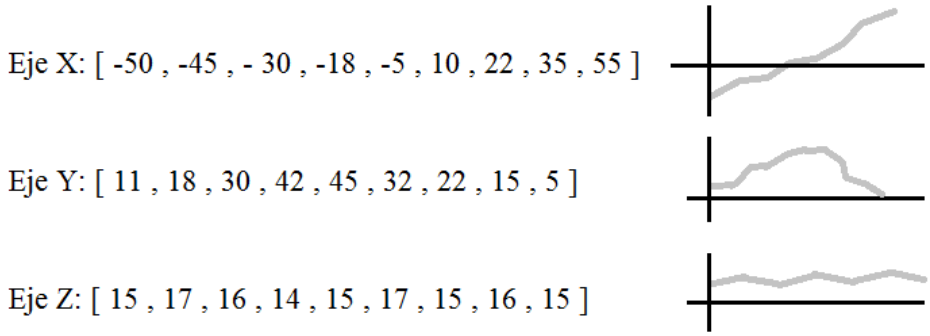



**Figura 35: Ejemplo simple de los datos que podríamos tener de la mano.**

Se puede ver que en la figura anterior, en el primer instante la mano está situada en la posición X=-50, Y=11 y Z=15.

Para saber si dicho movimiento es similar a otro, se pasaran como argumentos al algoritmo las diferentes secuencias de los ejes X Y Z de los dos gestos respectivamente.

Dicho algoritmo nos devuelve un resultado, siendo el coste de similitud. Cuanto más bajo sea dicho valor, más similares son los 2 gestos.

Se compararan las secuencias del eje X del gesto 1 con el gesto 2, las secuencias del eje Y, y las secuencias del eje Z. Si los tres costes que obtenemos son bajos, significa que los 2 gestos son similares, pero si algún coste es elevado, son diferentes.

### 4.6.2. El buffer.

Cuando se lanzara el reconocedor, se tendrá que ir comparando los gestos realizados con los gestos cargados en tiempo real. Como no podemos saber con exactitud en qué momento se empieza a realizar un gesto, usaremos un buffer para guardar los datos que nos llegan del Leap Motion, y que compararemos con los movimientos guardados.

El buffer estará constituido por la información de los *frames*, que son:

- Posición de la mano derecha en el eje X Y Z.
- Posición de la mano izquierda en el eje X Y Z.
- Velocidad en el eje X Y Z.

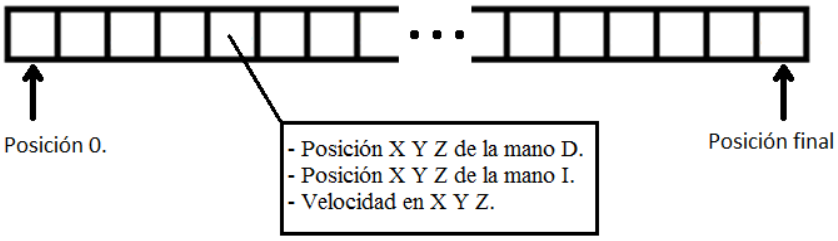



**Figura 36: Imagen abstracta del buffer.**

Cuando se haga un gesto para ser reconocido, se irán recogiendo los *frames* que devuelve el Leap Motion para guardarlos en el buffer. Mediante este último, ya podemos conseguir las secuencias que usaremos para el algoritmo.

El tamaño por defecto del buffer es igual a 35, pero el usuario podrá definir otro si lo desea. Cuando se llena, se usará la regla FIFO (*First In First Out*), por lo que borraremos el dato más antiguo, siendo el de la posición 0. Al eliminar dicha posición, se mueven los datos para dejar espacio en la última posición.

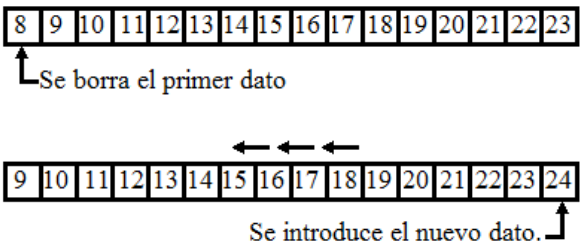



**Figura 37: Introducción de un nuevo dato al buffer.**

### 4.6.3. La estructura del reconocimiento.

Para hacernos una idea, en esta parte explicaré de forma muy breve los 2 elementos más importantes que forman el reconocedor:

- **El escuchador:**

El escuchador es la parte que recibe los datos del Leap Motion y que los introduce en el buffer. Después de añadir el nuevo dato al buffer, el escuchador se encargará de clonarlo e introducirlo a la cola de buffers. Dicha cola se usa para no sobrescribir los datos del buffer, y de este modo, no perder información.

- **El comparador:**

El comparador se ocupa de recoger, si lo hay, el buffer de la cola del escuchador, y comparar sus datos con los gestos cargados.

#### *4.6.3.1. Funcionamiento general del reconocimiento.*

En la siguiente figura podemos ver un resumen del funcionamiento del reconocedor:

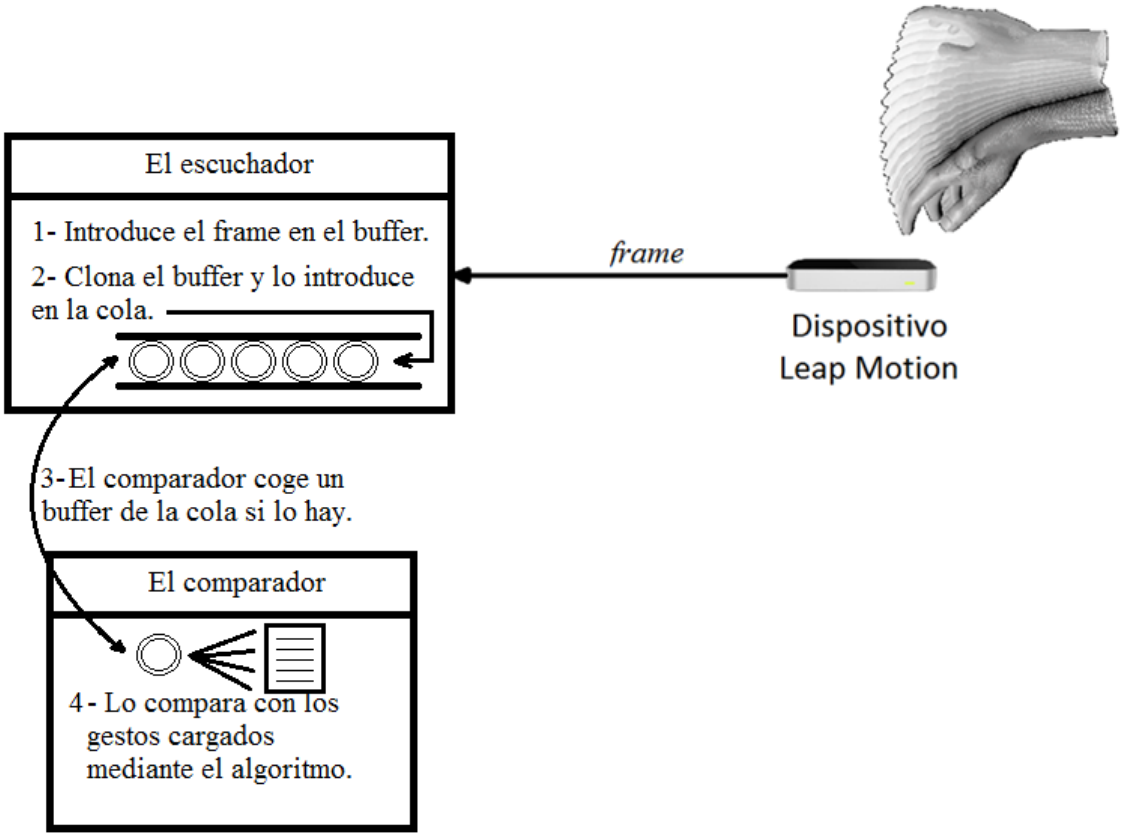



**Figura 38: Resumen del funcionamiento del reconocedor.**

### 4.6.4. El escuchador.

Al lanzar el reconocimiento, se crea el escuchador que tiene como rol principal recibir los datos del Leap Motion e introducirlos en el buffer.

Para empezar, crea el buffer que tendrá un tamaño por defecto igual a 35, pero que puede ser modificador por el usuario. Después, crea un proceso ligero en el que se ejecutara el comparador, pasándole como parámetros los gestos cargados y el coste de reconocimiento (Por defecto 1200, pero también modificable).

Al recibir un *frame* del Leap Motion, el escuchador:

1. Añade el *frame* al buffer.
2. Clona el buffer.
3. Añade a la cola de buffers el buffer clonado.
4. Envía un evento al comparador para avisarle de que hay buffers en la cola.

En este proceso existen 2 secciones críticas. Esto significa que hay una posibilidad en el que un dato pueda ser sobrescrito, y de esta forma perder información.

La primera está en el que el escuchador pueda recibir más *frames* de los que procesa. Si esta situación ocurre, se podrían añadir varios *frames* al buffer al mismo tiempo, y por lo tanto sobrescribirlo. Para ello se creó un semáforo en el que solo un recurso puede entrar a la sección crítica del paso 1 y 2.

La segunda sección crítica se sitúa en el momento en el que se añade el buffer clonado a la cola, ya que al mismo tiempo el comparador puede estar accediendo a la ella. Estas dos partes se controlan mediante un semáforo para no estar utilizando la cola de buffers a la vez.

Cuando se para el reconocimiento, se destruye el proceso ligero Comparador y se elimina el escuchador.

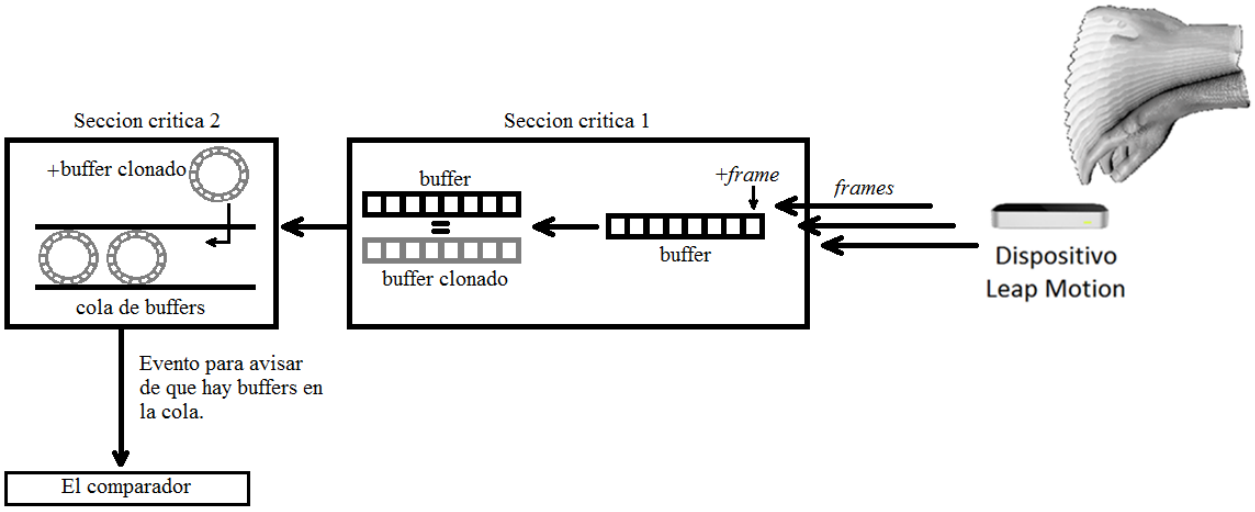



**Figura 39: Funcionamiento del escuchador para el reconocedor.**

### 4.6.5. El comparador.

El comparador sigue una serie de pasos para analizar el gesto realizado con los gestos cagados anteriormente:

#### *4.6.5.1. Extrae el buffer de la cola de buffers:*

Cuando el comparador recibe el evento del escuchador para avisarle de que hay un nuevo buffer en la cola, este extrae el buffer más antiguo (FIFO).

Esta operación está controlada mediante un semáforo que no permite a varios recursos acceder a la cola.

#### *4.6.5.2. Inicializa los datos del buffer:*

A continuación el comparador prepara los datos del buffer obtenido, sacando de él las secuencias que se compararán con las de los gestos cargados anteriormente, de las cuales hay:

- Una secuencia por cada eje con las posiciones de la mano derecha.
- Una secuencia por cada eje con las posiciones de la mano izquierda.
- Una secuencia por cada eje con su velocidad.

Aunque solo contemplamos la velocidad en un solo eje, tenemos que guardar las 3 diferentes secuencias del gesto realizado ya que se pueden tener diferentes movimientos cargados que dependen de la velocidad en ejes distintos.

Con los datos del buffer listos, se empieza a comparar con la lista de gestos grabados.

#### *4.6.5.3. Compara el gesto realizado con cada gesto cargado.*

El comparador entra en un bucle para analizar mediante el algoritmo DTW el buffer (gesto realizado) con cada gesto grabado anteriormente.

1. Se compara mediante el algoritmo las secuencias del gesto realizado con las secuencias del gesto cargado de la mano derecha.
2. Se compara mediante el algoritmo las secuencias del gesto realizado con las secuencias del gesto cargado de la mano izquierda.
3. El algoritmo DTW nos devuelve los diferentes costes de reconocimiento:
    a. Si algún coste de reconocimiento es mayor que el máximo permitido (por defecto 1200):
        i. Si hay más gestos en la lista por comparar, volvemos al paso 1.
        ii. Si no hay más gestos, se sale del bucle, y se espera comparar otro buffer de la cola.
    b. Si todos los costes cumplen los requisitos, pasamos al siguiente punto.

#### 4.6.5.4. Verifica la velocidad.

Si el gesto cargado tiene en cuenta la velocidad en algún eje, se calcula la velocidad media del movimiento realizado en dicho eje.

Como para el grabador de gestos, la velocidad media se calcula a partir y hasta un *frame* especifico. El usuario puede modificar los valores de dichos *frames*, pero por defecto el *frame* inicial es 10 y el *frame* final es 20.

#### 4.6.5.5. Verifica los ejes en los que el movimiento no participa.

Si en algún eje el movimiento tiene que ser constante, se verifica que los valores de dicha secuencia no varían mucho. Para ello se busca el valor mayor y menor de dicha secuencia para calcular su diferencia.

Si la diferencia es mayor a lo permitido, por defecto 60, el gesto realizado no cumple las condiciones. Dicho límite puede ser modificado por el usuario.

#### 4.6.5.6. Se reconoce el gesto.

Si todas las condiciones anteriores se cumplen, el gesto realizado es igual que el gesto cargado.

Los siguientes buffers que se analizaran tendrán algunos datos del buffer reconocido, por lo que para no reconocer el mismo gesto varias veces, se descartan todos los buffers que llegan con algún dato similar al reconocido.

## 4.7. La interfaz gráfica.

Para facilitar al usuario el uso del prototipo se diseñó una interfaz gráfica, que se compone de diferentes ventanas.

### 4.7.1. La pantalla principal.

Al ejecutar la aplicación, nos aparece la pantalla principal:

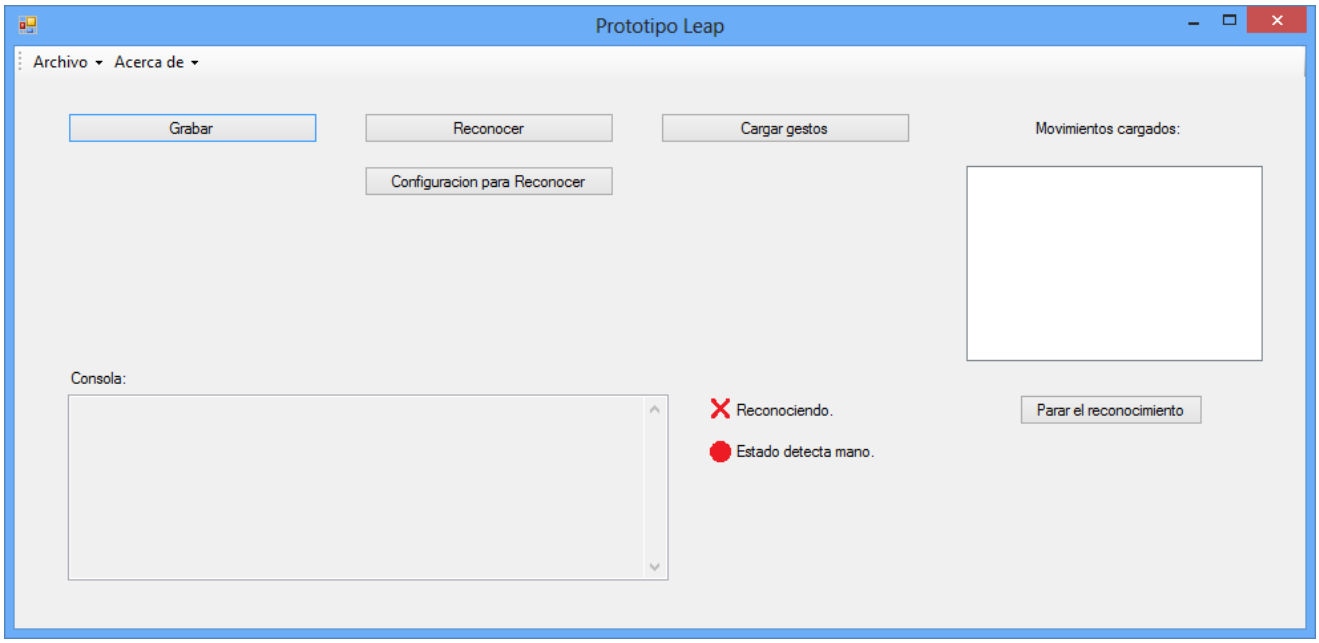



**Figura 40: Ventana principal del prototipo.**

Para no hacerlo complejo, decidí poner los botones que realizan las diferentes funciones del prototipo en la pantalla principal, que son:

- Grabar un gesto.
- Reconocer los gestos que se hacen.
- Cargar los gestos.

También se añadió un botón para configurar el reconocedor, y un cuadro en el que se verán los gestos cargados. Además, se introdujo una consola para tener un *feedback*, y 2 iconos para indicar el estado del reconocimiento y de si el Leap Motion detecta la mano. Todo ello se explicara en los siguientes apartados.

### 4.7.2. La creación de un gesto.

Para crear un gesto, el usuario pulsa el botón 'Grabar' de la pantalla principal, y aparece la siguiente:

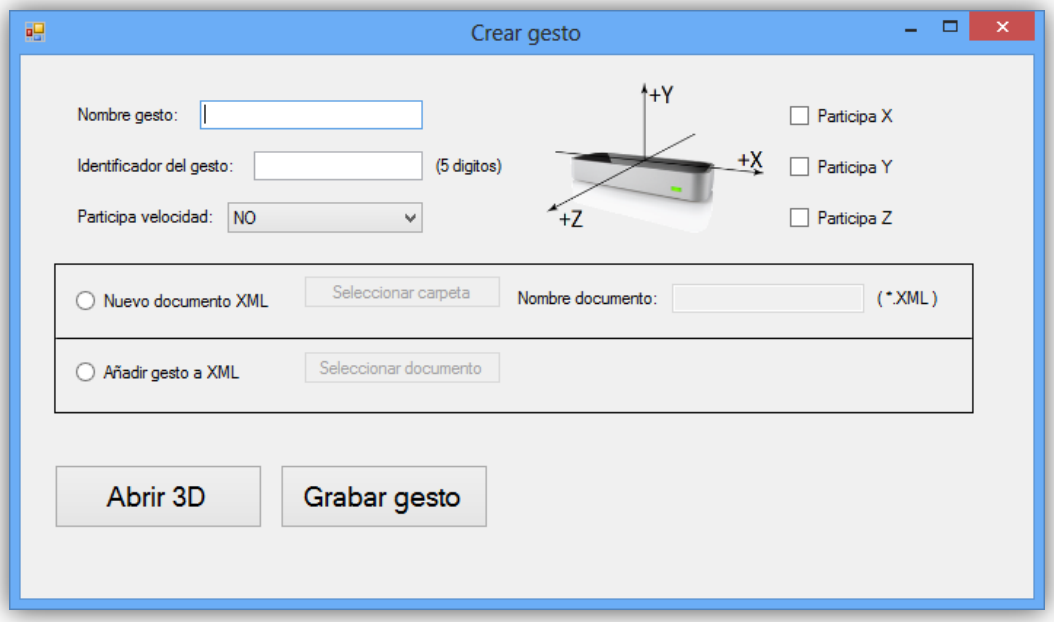



**Figura 41: Ventana para grabar un gesto.**

En dicha ventana se introduce los datos necesarios para la creación del gesto, que son: el nombre del gesto, un identificador de 5 dígitos, el eje en el que se quiere tener en cuenta la velocidad (si no se quiere considerar, se selecciona 'NO') y los ejes en los que participa el movimiento.

Este nuevo gesto puede guardarse en un documento XML ya existente (por lo tanto con uno o varios gestos), o crear un nuevo XML para dicho gesto.

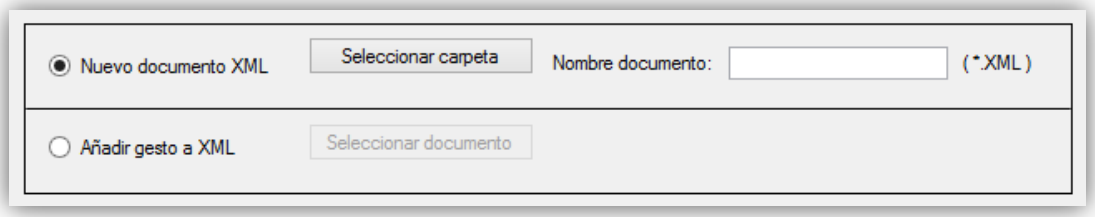



**Figura 42: Opciones para guardar un gesto.**

Cuando se selecciona una de las dos opciones, se activan sus campos relativos, y al seleccionar una carpeta, la ruta aparece debajo del botón para seleccionarlo.

Para poder ver el gesto en tiempo real, podemos activar el visualizador [19] mediante el botón ‘Abrir 3D’. Esta es una implementación libre en HTML y JavaScript para visualizar lo que ve el Leap Motion.

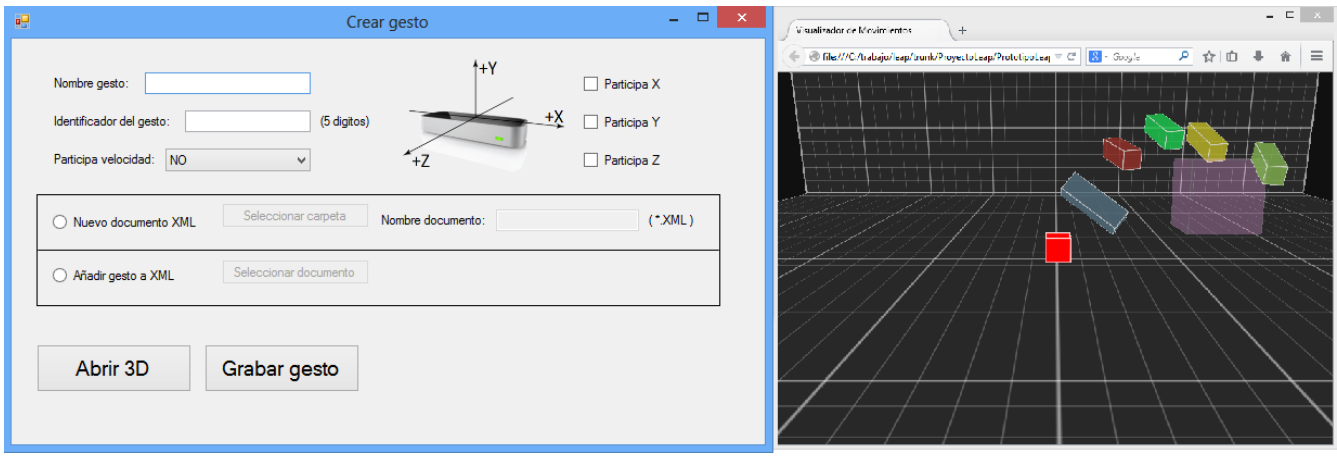



**Figura 43: El visualizador para la creación del gesto.**

Al lanzar la grabación, el programa verifica si los datos se han introducido correctamente:

- Que el campo ‘Nombre gesto’ no este vacío.
- Que el identificador sean 5 dígitos.
- Que se eligió al menos 1 eje en el que participa el movimiento.
- Que se eligió donde guardar el gesto:
    - Si se seleccionó ‘Nuevo documento XML’, se verifica que se seleccionó una carpeta y que se introdujo un nombre al documento.
    - Si se seleccionó ‘Añadir gesto a XML’, se verifica que se seleccionó un documento XML y que el identificador no este repetido.

Si alguna de estas condiciones no se cumplen, se muestra un mensaje de error.

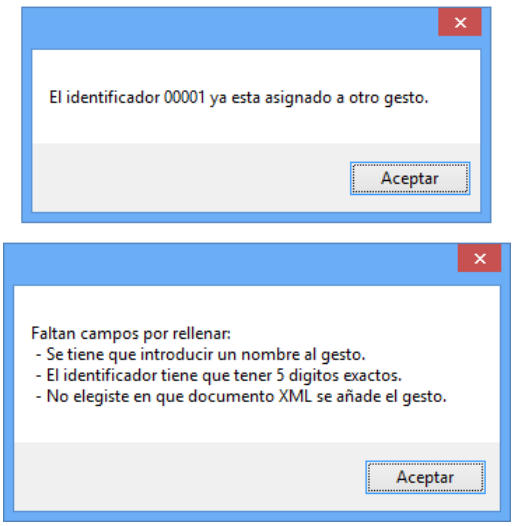



**Figura 44: Ejemplos de mensajes de error al grabar un gesto.**

Cuando se lanza la grabación, aparece una consola con un contador para avisarnos en qué momento se empieza a grabar, para que el usuario sepa en qué momento tiene que empezar a realizar su gesto.

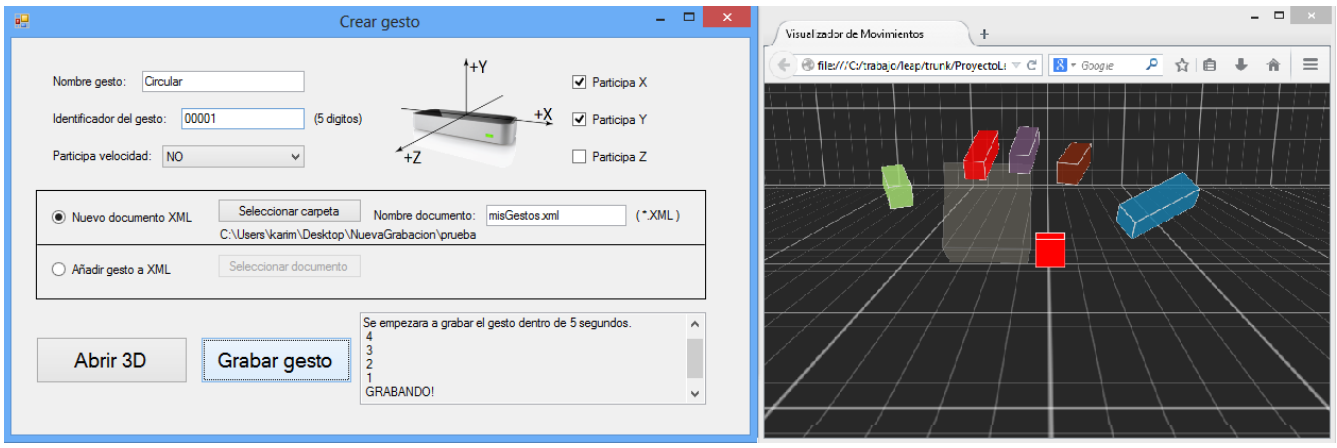



**Figura 45: Grabando un gesto.**

Mientras estamos realizando el gesto para su grabación, podemos ver en la consola el estado:

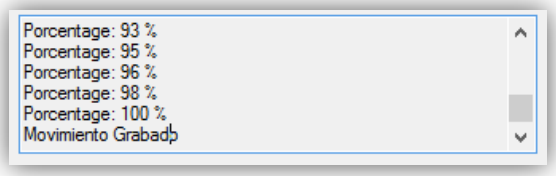



**Figura 46: Consola para la grabación, en el que vemos el estado.**

Al finalizar la grabación, dicho gesto se guarda en la ruta elegida anteriormente.

### 4.7.3. Carga de los gestos.

Para reconocer un gesto, primero hay que cargar la lista de gestos, por lo que al hacer clic en el botón 'Cargar' de la ventana principal, seleccionamos el documento XML con nuestros gestos.

En la parte derecha podemos ver los movimientos que contiene el XML:

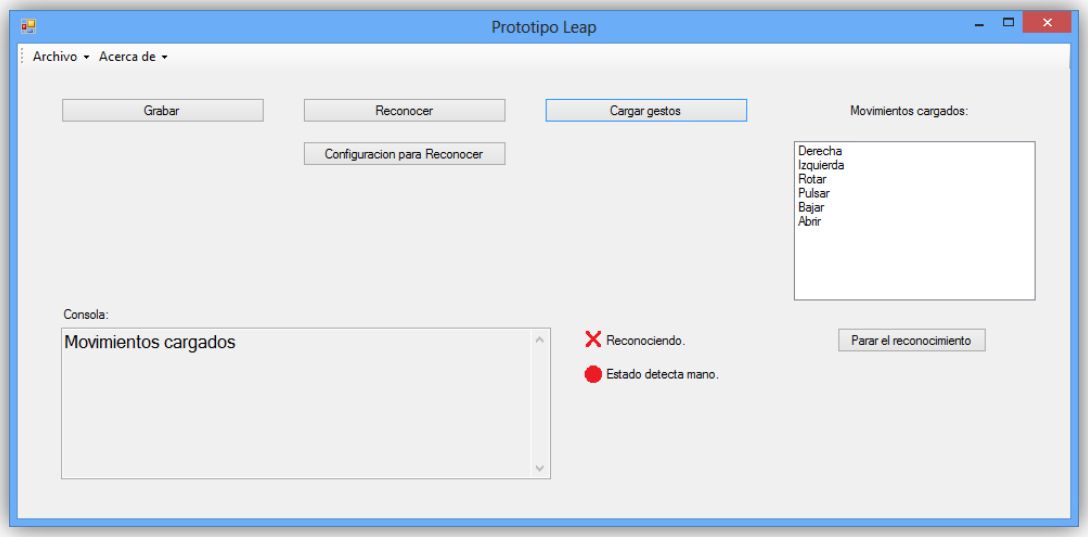



**Figura 47: Ventana en el que se ven los gestos cargados.**

El documento cargado como ejemplo consta de 6 movimientos: Derecha, Izquierda, Rotar, Pulsar, Bajar y Abrir.

Al seleccionar un gesto de la lista, podemos obtener su respectiva información:

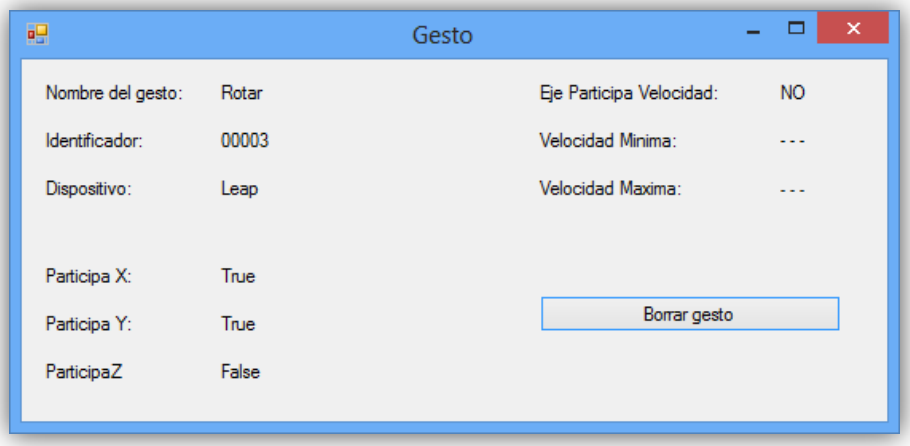



**Figura 48: Información del gesto.**

Además, el gestor nos permite borrar el gesto.

### 4.7.4. Reconocimiento de los gestos.

Después de haber cargado los gestos ya se puede empezar con el reconocimiento. Para ello, se hace clic en el botón 'Reconocer'.

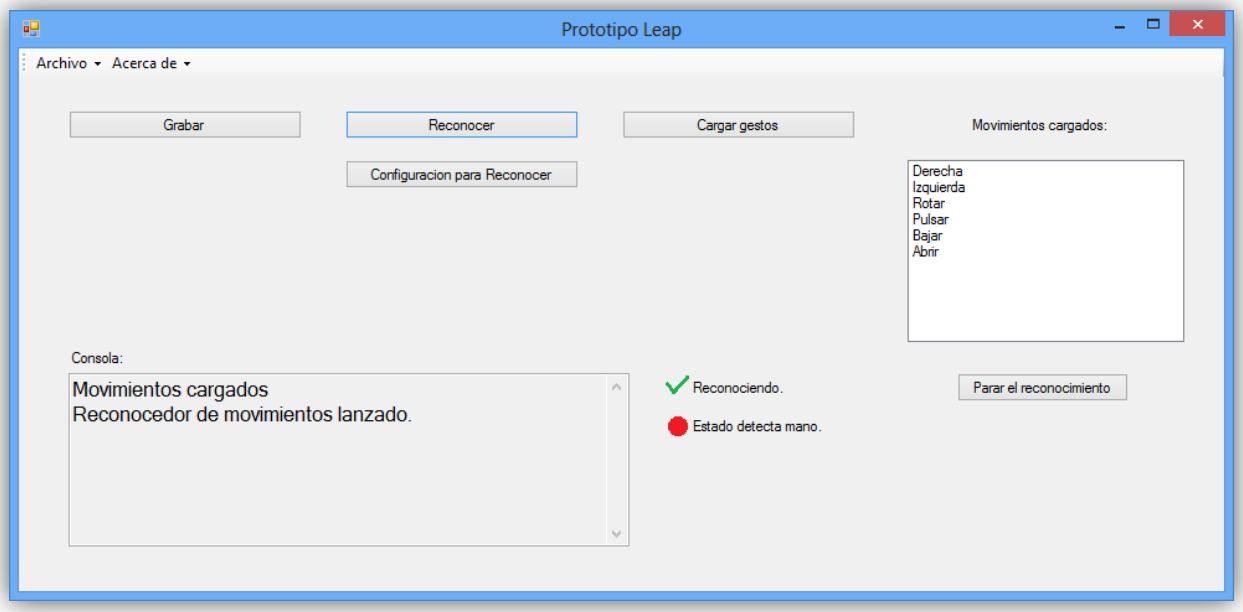



**Figura 49: Después de cargar los gestos, se activa el reconocedor.**

Cuando se está reconociendo los gestos, el icono 'Reconociendo' cambia:

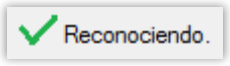



**Figura 50: El reconocedor está activado.**

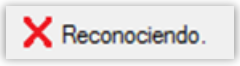



**Figura 51: El reconocedor esta desactivado.**

Para que el usuario sepa si el Leap Motion está detectando sus manos, puede verlo en 'Estado detecta manos':

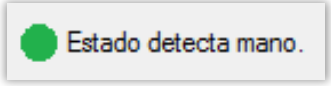



**Figura 52: El Leap Motion detecta la mano del usuario.**

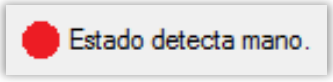



**Figura 53: El Leap Motion no detecta nada.**

Al realizar un gesto que está en la lista de gestos cargados anteriormente, aparece un mensaje abajo a la derecha:

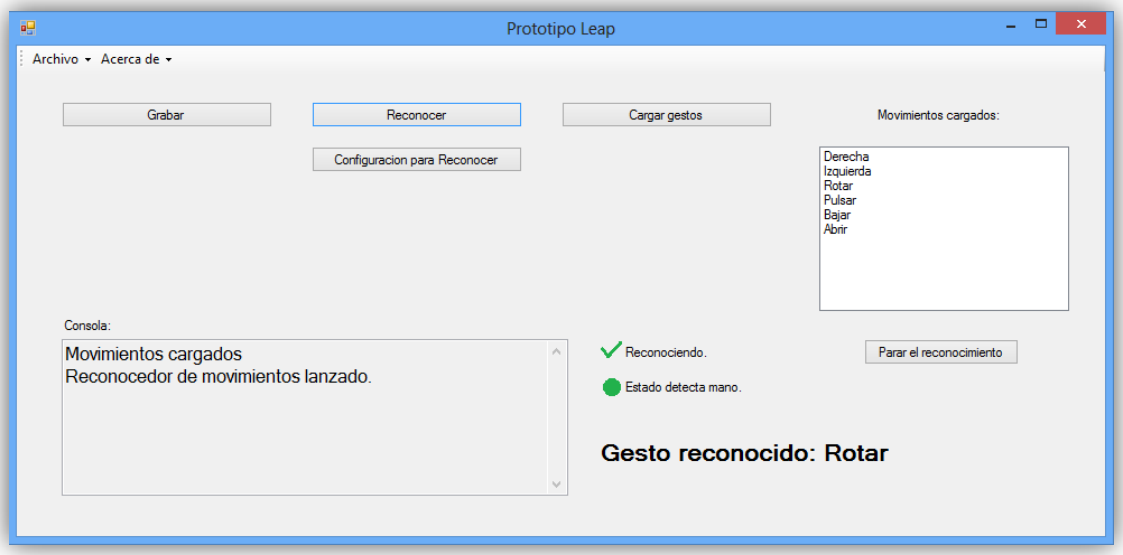



**Figura 54: Se reconoce el gesto 'Rotar'.**

También podemos configurar el reconocimiento al hacer clic en el botón ‘Configuración para Reconocer’, situado en la ventana principal.

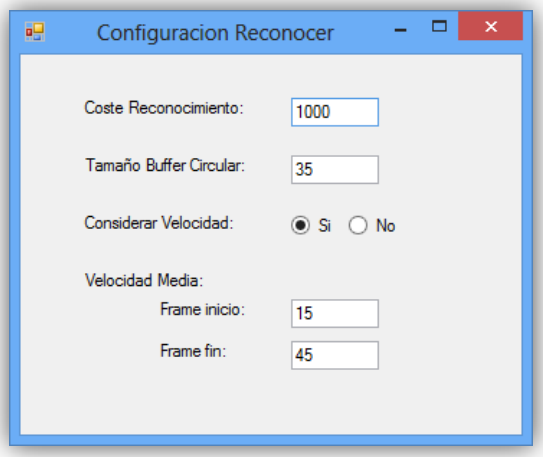



**Figura 55: Configuración del reconocedor.**

En ella se puede modificar:

- El coste del reconocimiento.
- El tamaño del buffer.
- Si se considera la velocidad.
- El *frame* de inicio y de fin para el cálculo de la velocidad media.

# 5. PRUEBAS Y RESULTADOS

## 5.1. Pruebas.

Para analizar el prototipo, se hicieron una secuencia de pruebas. Se redactó una lista de tareas que realizaran una serie de personas para probar el funcionamiento del prototipo.

Antes de empezar la realización de las pruebas, es necesario explicar a los usuarios las distintas funcionalidades y objetivos del prototipo.

### 5.1.1. Prueba 1: Grabación.

La primera tarea consiste en grabar nuevos gestos y añadirlos a un archivo XML ya creado que contiene gestos anteriormente grabados. Los usuarios no tardaron en localizar el botón que tiene la opción de crear un gesto nuevo dado que en la ventana principal se encuentran la mayoría de las funcionalidades importantes.

### 5.1.2. Prueba 2: Gestor.

La segunda tarea consiste en cargar el archivo XML que contiene los nuevos gestos que el usuario ha grabado en la prueba anterior en el reconocedor.

Después de cargarlos, se pidió a los usuarios acceder a la información de un gesto, y borrarlo del archivo XML.

### 5.1.3. Prueba 3: Reconocimiento.

La última tarea consiste en lanzar el reconocedor y realizar los diferentes gestos grabados en la prueba 1. De esta forma el prototipo reconocerá los gestos.

### 5.1.4. Pruebas para el juego educativo.

Como una de las finalidades del proyecto ha sido la incorporación del dispositivo Leap Motion al juego educativo desarrollado el semestre anterior (Ver apartado 2.1), definí los diferentes gestos que se usaran para interactuar con el juego.

Grabe los diferentes movimientos que tendrán asociados una acción en el juego para probar si al reproducirlos se reconocían fácilmente. Los gestos son:

- El gesto Derecha: Moviendo la mano de izquierda a derecha.
- El gesto Izquierda: Moviendo la mano de derecha a izquierda.
- El gesto Pulsar: Moviendo la mano hacia la pantalla.
- El gesto Bajar: Moviendo la mano de arriba abajo.
- El gesto Rotar: Moviendo la mano de forma circular.
- El gesto Abrir: Juntando las dos manos y separándolas.

## 5.2. Resultados.

Al realizar las pruebas con los usuarios, pude observar diferentes situaciones, de las cuales las más importantes:

- Al abrir la ventana para grabar un nuevo movimiento, pocos se atrevieron a darle clic al botón 'Abrir 3D', ya que no sabían exactamente para que se usa. Una mejora sería integrar el visualizador en la ventana. Hablare de este cambio en el apartado 'Futuras líneas de desarrollo'.

- Para saber que eje participaba en el movimiento, se les tuvo que explicar detalladamente las diferencias que existían al seleccionar un eje.

- Cuando un usuario se equivocaba en rellenar los datos, aparecía un mensaje describiendo el error, por lo que podían corregirlo fácilmente.

- En la segunda prueba, los usuarios no sabían cómo acceder a los datos del gesto, dado que es necesario hacer doble clic en el nombre del gesto. Para ello se podría añadir un apartado de ayuda.

- La tercera prueba no tenía dificultad, y los usuarios consiguieron realizar de nuevo los movimientos anteriormente grabados para ser reconocidos.

- El haber integrado una interfaz gráfica al prototipo ayudo a que su manejo sea bastante fácil e intuitivo, siendo el periodo de aprendizaje corto.

# 6. CONCLUSIONES

## 6.1. Conclusión.

Con el desarrollo de este proyecto he podido descubrir un nuevo dispositivo llamado Leap Motion. Este último facilita la interacción entre persona y ordenador mediante gestos intuitivos que se realizan en el aire con nuestras manos, ofreciendo infinitas posibilidades al usuario.

A lo largo del semestre he ido aprendiendo como desarrollar programas que usan dicho dispositivo, y conseguí realizar las tareas propuestas y cumplir con los objetivos planteados.

El prototipo implementado ofrece a los desarrolladores la posibilidad de integrar el dispositivo Leap Motion en cualquier aplicación mediante la creación de gestos que posteriormente serán reconocidos.

Mediante las diferentes pruebas realizadas pude observar que el prototipo es intuitivo, funciona correctamente y cuenta con las validaciones suficientes para evitar errores por parte de los usuarios.

A pesar de que el dispositivo está al alcance de todos por su bajo precio en el mercado, todavía no ha recibido el suficiente apoyo por parte de los desarrolladores, y por lo tanto a día de hoy ofrece pocas aplicaciones.

Sin embargo, creo que este dispositivo innovador tiene muchas posibilidades de desarrollo y un futuro prometedor. Además, cabe destacar que ya existen ordenadores [20] en el mercado que integran dicha tecnología.

# 7. FUTURAS LINEAS DE DESARROLLO

## 7.1. Mejoras al proyecto.

Durante este proyecto se siguió el plan de trabajo y se realizaron las diferentes tareas para la implementación del prototipo, sin embargo, se pueden hacer diferentes modificaciones y añadir algunas funcionalidades para mejorarlo. Aquí describo algunas de ellas.

### 7.1.1. Grabar un video al crear el movimiento.

Para grabar un gesto tenemos la opción de abrir el visualizador. Esta se visualiza con el navegador predeterminado del usuario, pero uno de los problemas principales es la compatibilidad.

Para no depender del navegador del usuario al grabar un gesto, se puede integrar dicha parte en la ventana del proyecto:

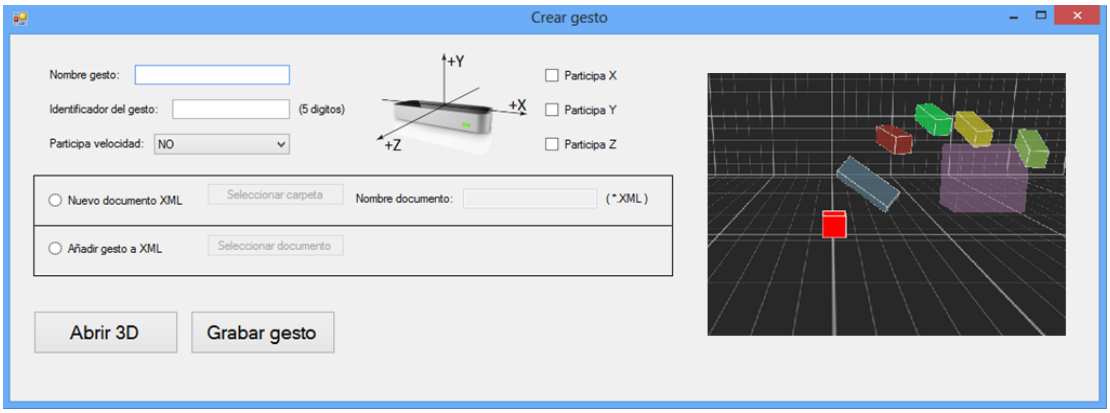



**Figura 56: Ejemplo de la integración del visualizador en la ventana para crear gestos.**

Otra mejora importante en este tema, es grabar un video de dicho movimiento en tres dimensiones, para que el usuario sepa exactamente que gesto se grabó. Además, se podría reproducir el video del gesto al cargarlo.

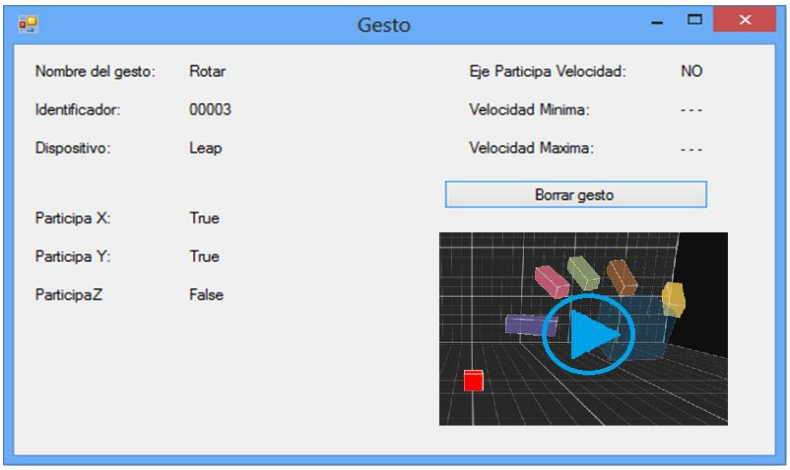



**Figura 57: Ejemplo para la reproducción del video.**

### 7.1.2. Mejoras del gestor.

El gestor de gestos sigue teniendo pocas funcionalidades ya que solo permite añadir o borrarlo un gesto.

Una mejora interesante seria poder cambiar los datos del gesto después de haberlo grabado, como por ejemplo el nombre, el identificador, la velocidad mínima, la velocidad máxima o los ejes en los que participa el movimiento.

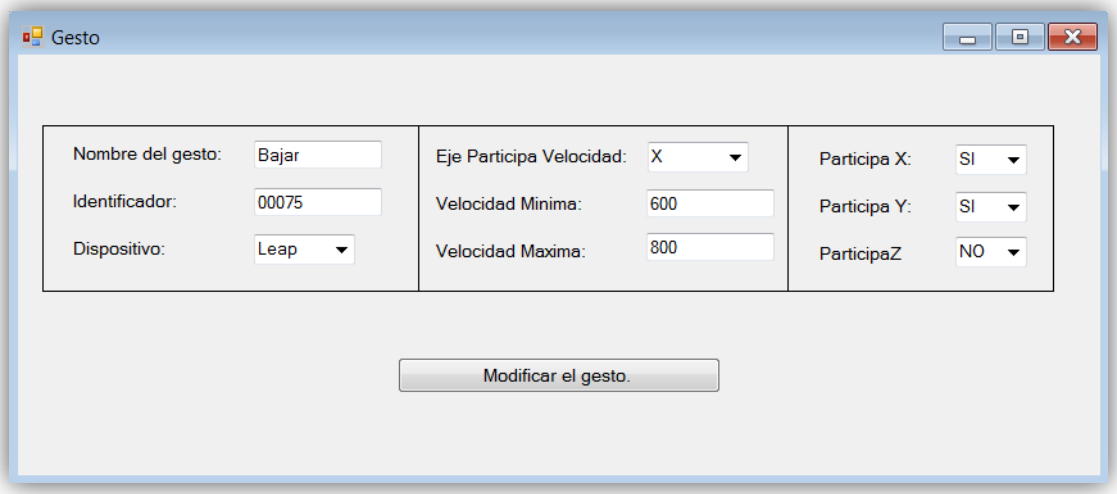



**Figura 58: Ejemplo para la modificación de datos del gesto.**

También se podría añadir una funcionalidad para pasar un gesto de un documento XML a otro, y así facilitar importar y exportar gestos.

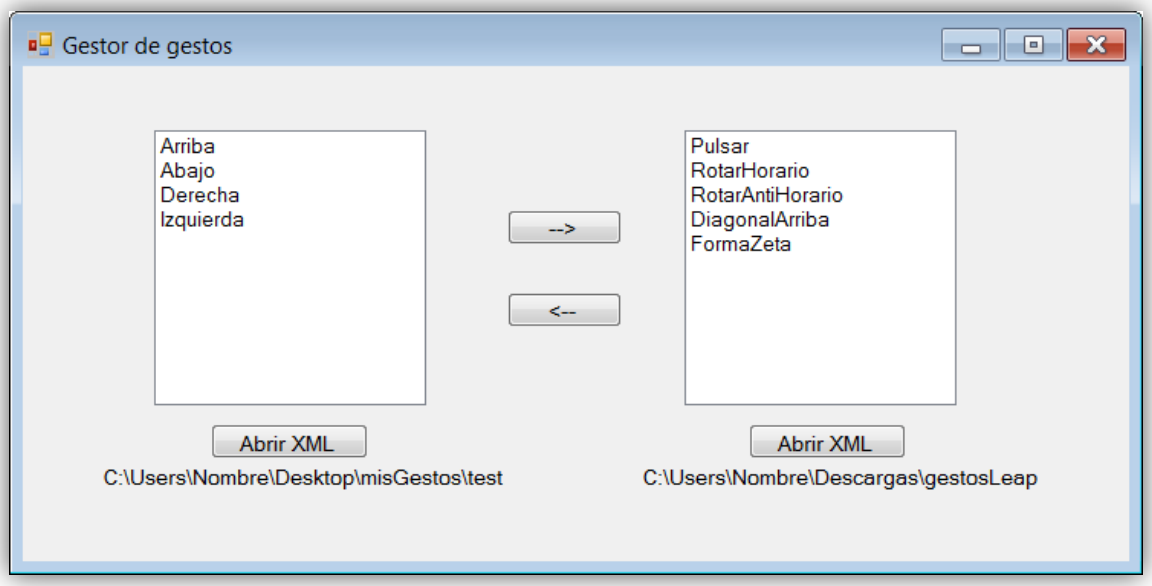



**Figura 59: Ejemplo para pasar los gestos de un documento a otro.**

## 7.2. Versión 2 del SDK.

Para este proyecto se usó la última versión del SDK en su momento, siendo la v.1.2.0.10970.

El 28 de mayo salió la versión 2 en beta [21], siendo una mejora de la anterior en casi todos los aspectos, y ofreciéndonos muchos más datos con más precisión.

A continuación describo las diferentes novedades:

### 7.2.1. Juntando dedos: El pellizco.

En la versión 1, al tocar un dedo con otro, el Leap Motion no era capaz de diferenciarlos, por lo que se perdía la información de estos 2 dedos como lo podemos ver en la siguiente figura.

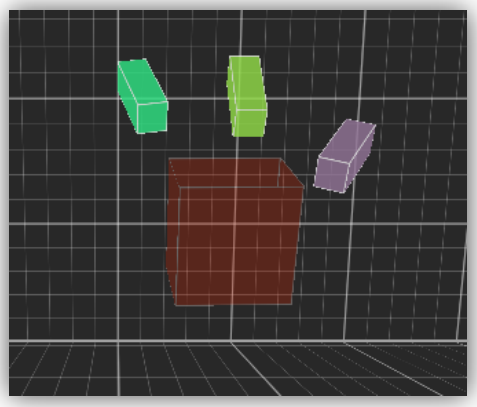


**Figura 60: Juntando dos dedos con la versión 1.**

Sin embargo, con la versión 2, obtenemos los datos precisos del gesto, y con los diferentes huesos que forman el dedo.

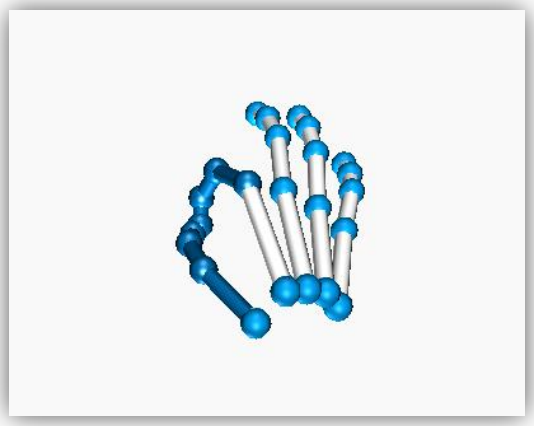


**Figura 61: Juntando dos dedos con la versión 2.**

### 7.2.2. Agarrar o cerrar el puño.

Otro problema de la versión 1 es que al cerrar la mano, desaparecían los dedos, y el Leap Motion era incapaz de verlos.

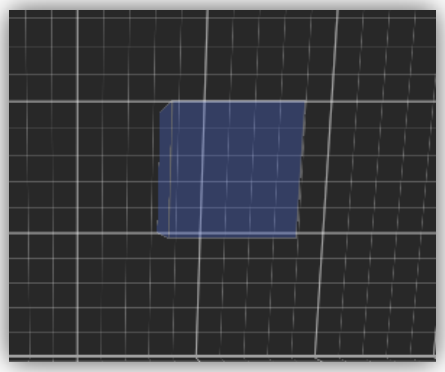


**Figura 62: Cerrando el puño en la versión 1.**

A diferencia de la versión 1, la versión 2 consigue obtener los datos de cada dedo de forma muy precisa.

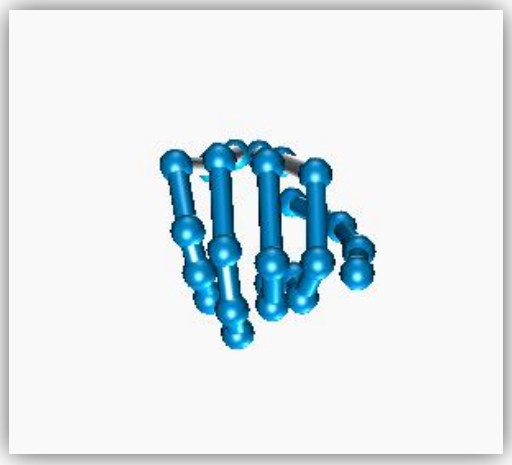


**Figura 63: Cerrando el puño con la versión 2.**

### 7.2.3. Probabilidades y estimación.

Una de las mejoras importantes fue el cálculo de la estimación de los miembros de la mano. Si se pone una mano encima de otra, al estar el dispositivo Leap Motion debajo, le es muy complicado ver lo que hay encima de la primera mano.

En la primera versión al acercar una mano con la otra perdíamos información de algunos miembros. Dicho resultado se puede ver en las siguientes imágenes:

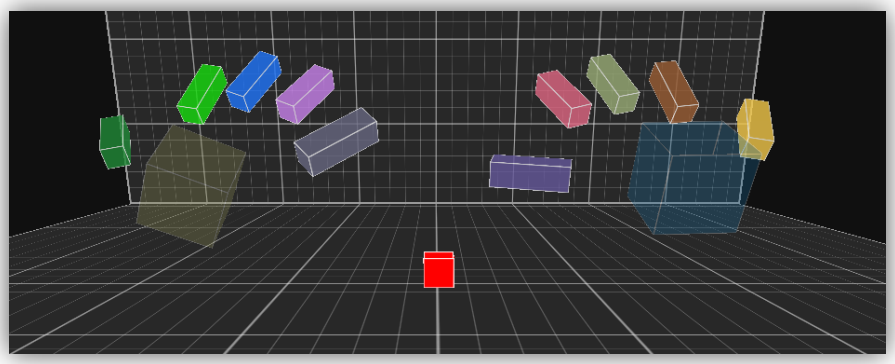


**Figura 64: Visualización de las 2 manos en la versión 1.**

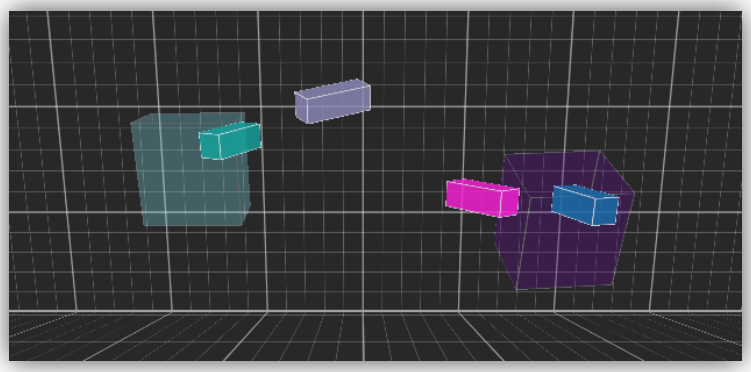


**Figura 65: Superponiendo las manos una encima de otra en la versión 1.**

Con la versión 2, el Leap Motion intenta estimar la posición de cada miembro para no perder información.

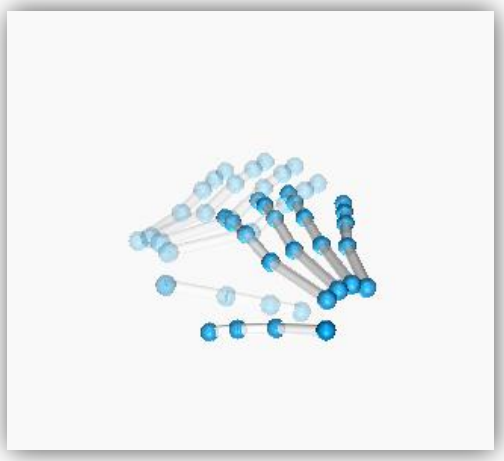


**Figura 66: Superponiendo las manos en la versión 2.**

### 7.2.4. Posición y rotación de cada esqueleto del dedo.

Esta nueva versión, aunque por ahora este en fase de pruebas antes de su lanzamiento oficial, ofrece información muy detallada de cada miembro.

Se puede obtener por cada hueso de cada dedo su posición exacta y su ángulo de rotación.

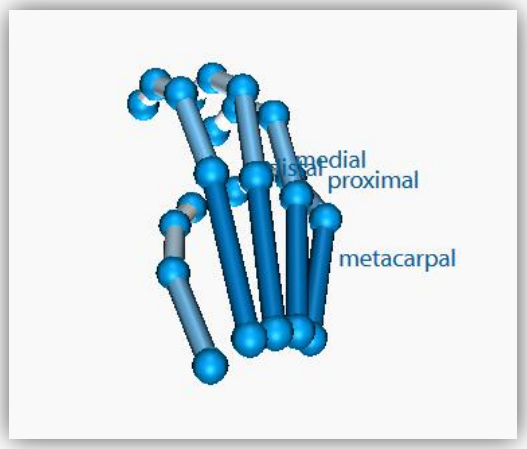



**Figura 67: Información de los huesos del dedo.**

### 7.2.5. Diferenciación de las manos.

De momento en el prototipo se supone que la mano más a la derecha es la mano derecha, y la que está más a la izquierda es la mano izquierda. El problema es que cuando por ejemplo se cruzan las manos, y la mano derecha pasa a estar más a la izquierda, y vice versa, el prototipo no lo detectaría bien.

Sin embrago, ahora el Leap Motion también puede diferenciar la mano derecha de la izquierda, independientemente de su posición.

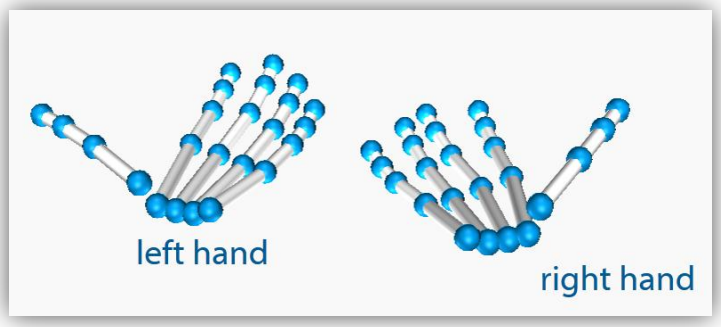



**Figura 68: Diferenciando las manos en la versión 2.**

### 7.2.6. Diferenciación de los dedos.

Anteriormente, al tener los datos de una mano, obteníamos una lista con la información de sus dedos. El problema era que no podíamos saber con exactitud de que dedo se trata.

Mediante la posición podíamos saber cuál era el que está situado:

- Más a la derecha: Su posición en el eje X es la mayor.
- Más a la izquierda: Su posición en el eje X es la menor.
- En el centro: Su posición en el eje Z es la mayor.

Pero no siempre podemos afirmar que el dedo que está más a la derecha sea el meñique y que el que está más a la izquierda sea el pulgar, ya que la mano podría estar al revés. Como consecuencia de este problema, al grabar un gesto con el prototipo, no guardamos los datos de cada dedo (Aunque en el esquema lo definimos, lo dejamos vacío.).

Sin embargo ahora no se tendría ninguna dificultad en grabarlos con la versión 2 del SDK, ya que el Leap Motion nos ofrece detalladamente la información de cada dedo, independientemente de la posición de la mano.

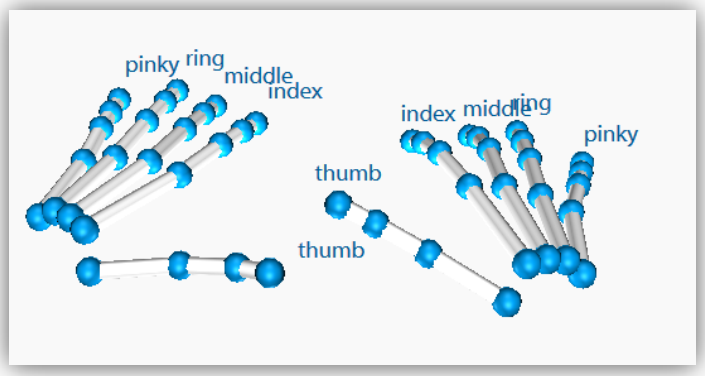



**Figura 69: Diferenciando los dedos de las manos con la versión 2.**

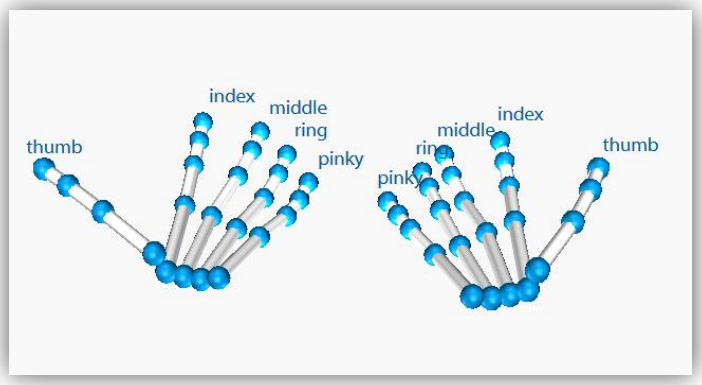



**Figura 70: Diferenciando los dedos con las manos al revés.**

# 8. BIBLIOGRAFÍA

# 8.1. Bibliografía

[1] Motion Control, «Leap Motion,» 2014. [En línea]. Available: https://www.leapmotion.com/. [Último acceso: 03 04 2014].

[2] «Openlab,» [En línea]. Available: http://openlab.com.au/shop/leap-motion/. [Último acceso: 03 05 2014].

[3] Wikipedia Community, «Wikipedia,» 5 Abril 2014. [En línea]. Available: http://en.wikipedia.org/wiki/Leap_Motion.

[4] «Insidehardware,» 29 Mayo 2014. [En línea]. Available: http://www.insidehardware.it/news/148-corporate-news/3808-leap-motion-annuncia-la-public-develper-beta-del-suo-sdk-2-0#.U42GCih7N-Y.

[5] Motion Control, «Leap Motion Airspace,» [En línea]. Available: https://airspace.leapmotion.com/categories/all. [Último acceso: 20 Mayo 2014].

[6] «AppBrain,» 20 Mayo 2014. [En línea]. Available: http://www.appbrain.com/stats/number-of-android-apps.

[7] R. O'Leary, «Github,» 13 Noviembre 2013. [En línea]. Available: https://github.com/roboleary/LeapTrainer.js/tree/master.

[8] D. Crockford, «JSON,» [En línea]. Available: http://www.json.org/. [Último acceso: 27 Febrero 2014].

[9] R. O'Leary, «Youtube,» 01 Septiembre 2013. [En línea]. Available: http://www.youtube.com/watch?v=JVqalPM9pHs.

[10] Epikur, «Sourceforge,» 05 Septiembre 2013. [En línea]. Available: http://sourceforge.net/projects/airkey/files/source/.

[11] T. Armour, «Github,» 20 Diciembre 2013. [En línea]. Available: https://github.com/jaanga/gestification/tree/gh-pages/cookbook/jest-play.

[12] Biovision. [En línea]. Available: http://www.character-studio.net/bvh_file_specification.htm. [Último acceso: 18 Febrero 2014].

[13] M. Akamatsu, «Github,» 03 Febrero 2013. [En línea]. Available: https://github.com/akamatsu/aka.leapmotion.

[14] M. Akamatsu, «Akamatsu,» [En línea]. Available: http://akamatsu.org/aka/max/objects/. [Último acceso: 15 Abril 2014].

[15] «LeapMotion Developer,» [En línea]. Available: https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html. [Último acceso: 05 Febrero 2014].

[16] «Psb.ugent,» [En línea]. Available: http://www.psb.ugent.be/cbd/papers/gentxwarper/index.htm. [Último acceso: 10 Abril 2014].

[17] «Psb.ugent,» [En línea]. Available: http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorithm.htm. [Último acceso: 10 Abril 2014].

[18] D. Oblak, «Github,» 10 Enero 2013. [En línea]. Available: https://github.com/doblak/ndtw.

[19] J. Garnier, «Github,» 24 Enero 2013. [En línea]. Available: https://github.com/juliangarnier/leapjs/tree/master/examples.

[20] «Amazon,» [En línea]. Available: http://www.amazon.es/HP-ENVY-17-j165es-Leap-Motion/dp/B00GMZSC48. [Último acceso: 27 Marzo 2014].

[21] Motion Control, «Leap Motion,» 28 Mayo 2014. [En línea]. Available: https://developer.leapmotion.com/.

[22] D. H. Michael Buckwald, «LeapMotion,» Motion Control, [En línea]. Available: https://www.leapmotion.com/. [Último acceso: 17 Abril 2014].

# A. ANEXO

# A.1. Manual de usuario.

En este manual explicaremos como realizar las diferentes funciones que nos ofrece el prototipo, de las cuales: grabar un movimiento, cargar una lista de gestos y lanzar el reconocimiento. Además se explicaran las posibles configuraciones que se pueden realizar.

## A.1.1. Grabación de un movimiento.

Para empezar a grabar un nuevo gesto, se hace clic en el botón 'Grabar' que se sitúa en la ventana principal de la aplicación.

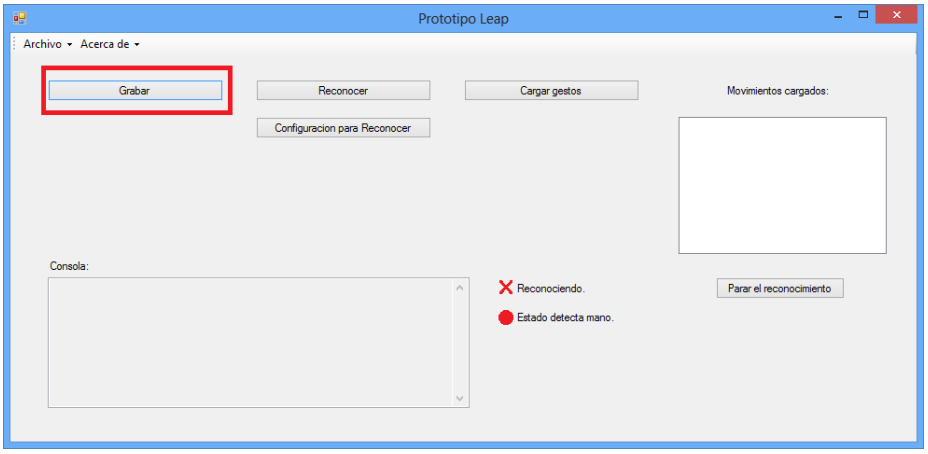



**Figura 71: Grabación de un nuevo gesto.**

En ese momento aparece una nueva ventana para la creación del gesto, en la que introducimos un nombre para el gesto y un identificador de 5 dígitos. Además seleccionamos el eje en el que se quiere tener en cuenta la velocidad, y los ejes en el que varía el movimiento.


**Figura 72: Datos para la grabación del nuevo gesto.**

Crearemos un movimiento de la mano de forma diagonal, empezando desde la esquina inferior izquierda hasta la esquina superior derecha. Lo llamaremos ‘ArribaDer’.

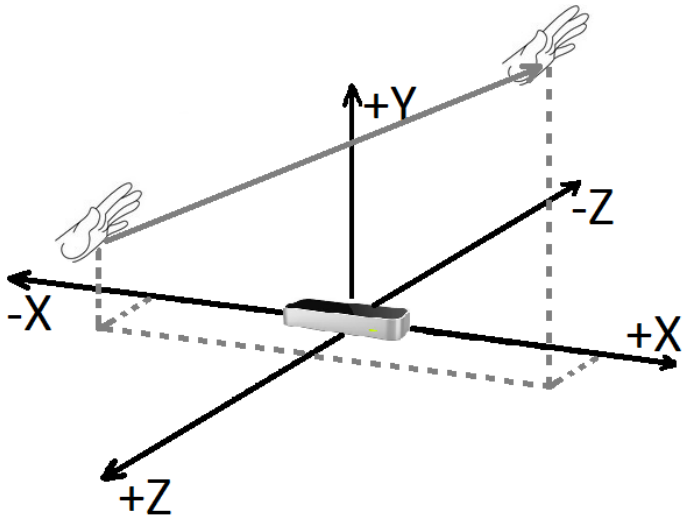



**Figura 73: Movimiento ‘ArribaDer’.**

El identificador será 00007, y tendremos en consideración la velocidad en el eje X. La profundidad tendrá que ser constante, pero dejaríamos al usuario elegir la que quiera.

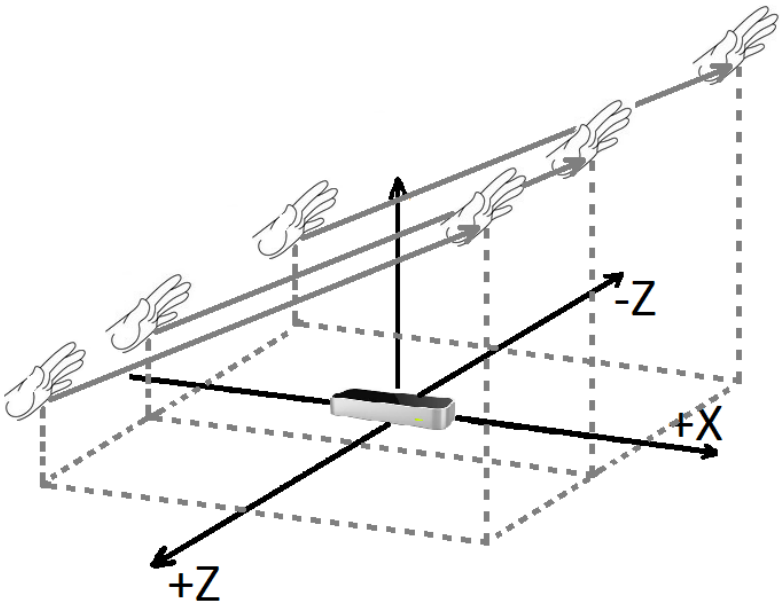



**Figura 74: Posibles movimientos para reconocer el gesto 'ArribaDer'.**

Añadiremos este nuevo gesto a un documento XML ya existente, por lo que hacemos clic en ‘Añadir gesto a XML’ y seleccionamos el documento.

En la siguiente figura podemos ver los datos que introducimos para crear el gesto ‘ArribaDer’.

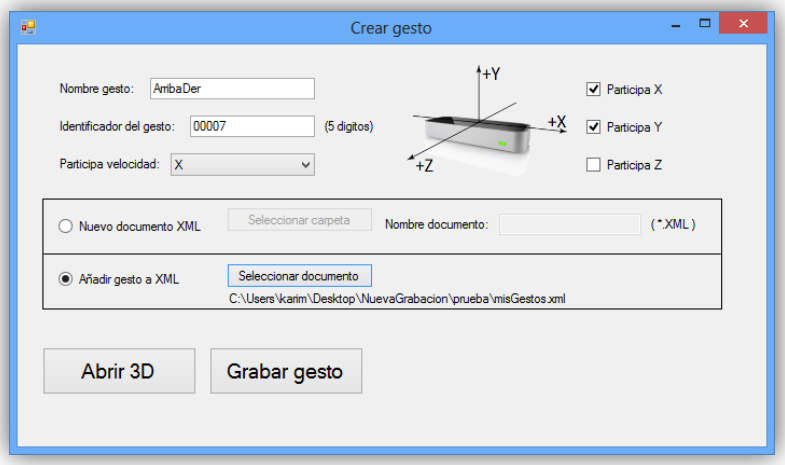
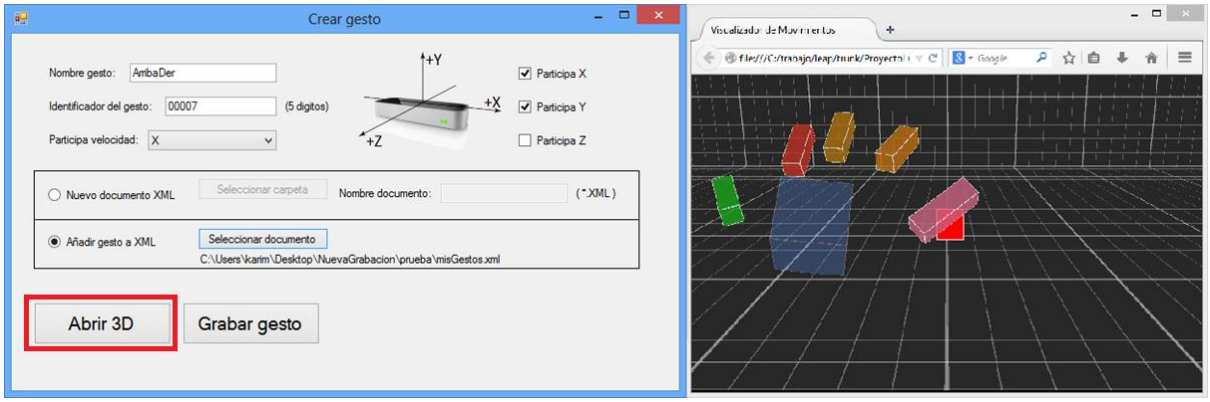



**Figura 75: Ejemplo para la creación del gesto 'ArribaDer'.**

Podemos darle al botón 'Abrir 3D' para poder visualizar nuestra mano, y asegurarnos de que todos los miembros sean visibles.




Para empezar a grabar el gesto, hacemos clic en 'Grabar gesto', y después de 5 segundos la grabación empieza.


**Figura 76: Pulsando el botón 'Grabar gesto'.**

Cuando el contador pasa a ser 0, se empiezan a guardar los datos del movimiento.

## A.1.2. Carga de un archivo XML.

Para cargar un documento XML, se hace clic en 'Cargar gestos', situado en la ventana principal de la aplicación, y posteriormente se selecciona el documento.

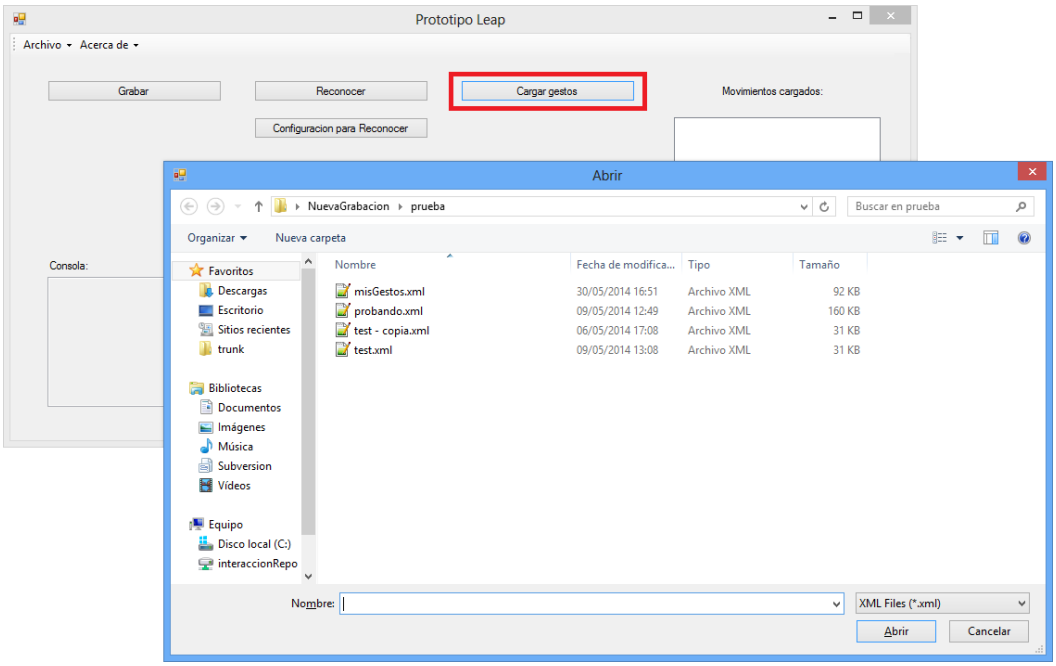



**Figura 77: Cargando un documento XML con varios gestos.**

Al cargar los gestos, aparecen sus nombres en la pantalla principal. En el ejemplo se puede ver que tenemos el gesto: Derecha, Izquierda, Rotar, Pulsar, Bajar, Abrir y finalmente el gesto que acabamos de añadir, ArribaDer.

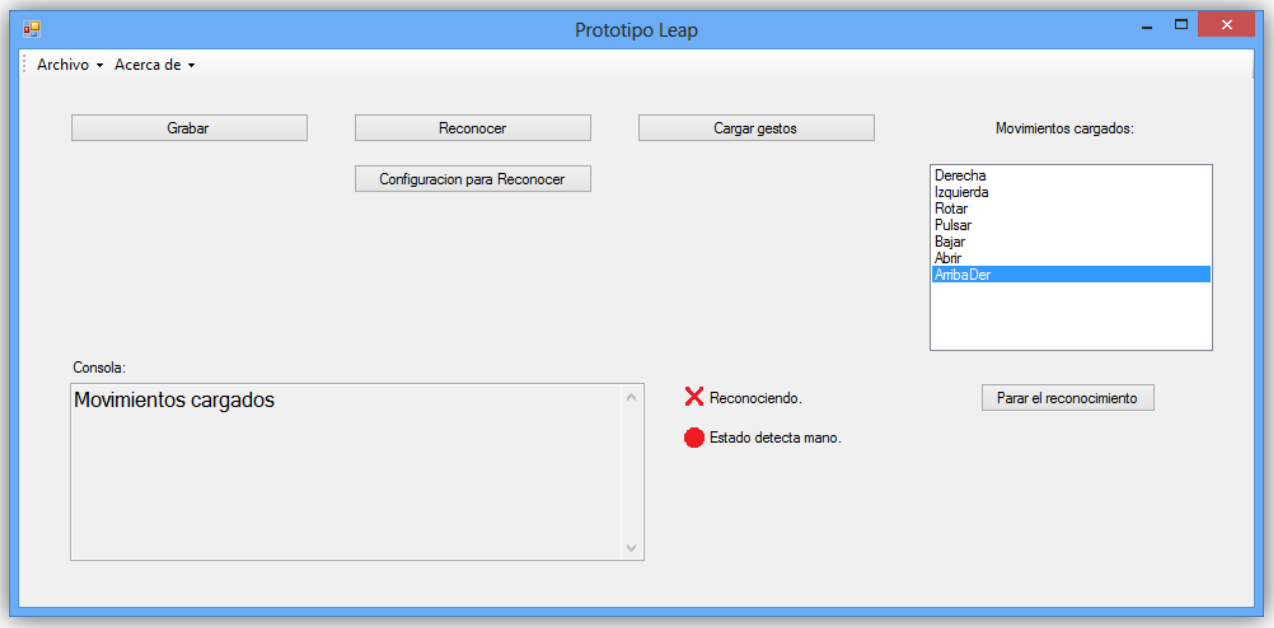



**Figura 78: Después de haber cargador los gestos.**

Para poder ver la información del gesto, se le hace doble clic.

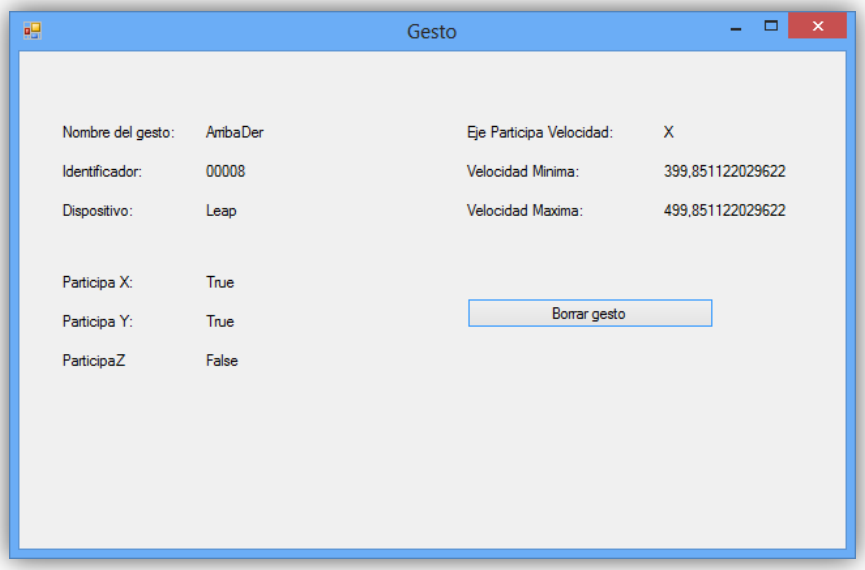



**Figura 79: Información del gesto 'ArribaDer'.**

En ella vemos su nombre, su identificador, los ejes en el que participa el movimiento y la velocidad mínima y máxima si se tiene en cuenta.

Si se quiere borrar el gesto, se hace clic en ‘Borrar gesto’.

### A.1.3. Configuración del reconocimiento.

Para modificar los parámetros del reconocedor, se hace clic en ‘Configuración Reconocer’, y aparece la ventana en la que se pueden cambiar los valores por defecto.

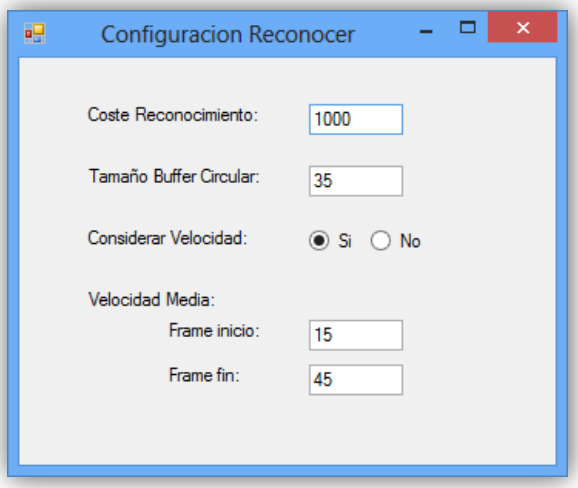



**Figura 80: Configuración del reconocedor.**

### A.1.4. Lanzar el reconocimiento.

Para lanzar el reconocimiento, se hace clic en 'Reconocer'. Podemos ver el icono 'Reconociendo' pasando de ser una cruz roja a un *check* verde. Cuando nuestra mano es detectada por el Leap Motion, el icono de 'Estado detecta mano' pasa a ser verde.

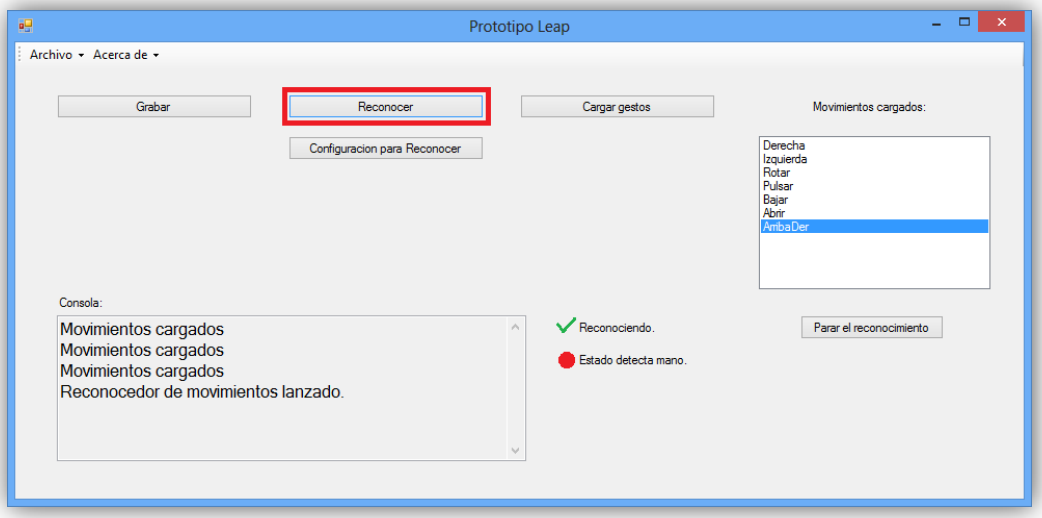



**Figura 81: Lanzando el reconocedor.**

Al realizar algún gesto de la lista que hemos cargado, se reconoce y aparece un mensaje en la esquina inferior derecha.

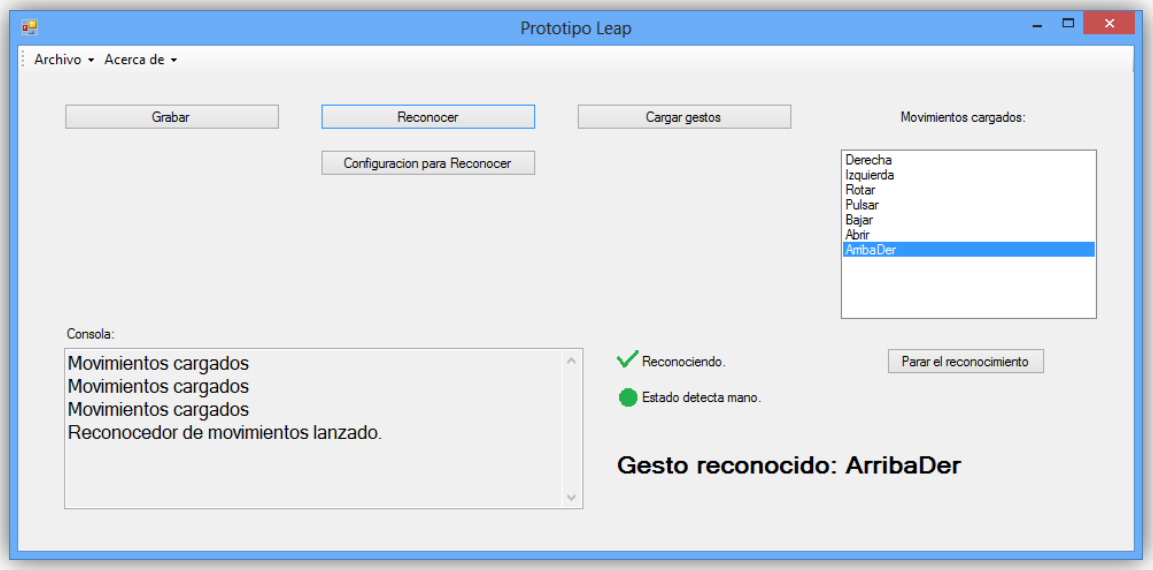



**Figura 82: Realizando el movimiento 'ArribaDer'.**

# A.2. Estructura de un documento con una lista de gestos.

## A.2.1 Esquema del documento.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="ColeccionGestos">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Gesto" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Nombre">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:minLength value="1"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="IdGesto">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:length value="5"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="Dispositivo">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value="Leap"/>
                    <xs:enumeration value=""/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="VelocidadMin">
                <xs:simpleType>
                  <xs:restriction base="xs:double">
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="VelocidadMax">
                <xs:simpleType>
                  <xs:restriction base="xs:double">
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="ParticipaVelocidad">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value="X"/>
                    <xs:enumeration value="Y"/>
                    <xs:enumeration value="Z"/>
                    <xs:enumeration value="NO"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
```

```
            <xs:element name="ParticipaX" type="xs:boolean"/>
            <xs:element name="ParticipaY" type="xs:boolean"/>
            <xs:element name="ParticipaZ" type="xs:boolean"/>
            <xs:element name="Esqueleto" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Joint" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="Nombre">
                          <xs:simpleType>
                            <xs:restriction base="xs:string">
                              <xs:enumeration value="Mano_Derecha"/>
                              <xs:enumeration value="Mano_Izquierda"/>
                              <xs:enumeration value="Dedo_D_1"/>
                              <xs:enumeration value="Dedo_D_2"/>
                              <xs:enumeration value="Dedo_D_3"/>
                              <xs:enumeration value="Dedo_D_4"/>
                              <xs:enumeration value="Dedo_D_5"/>
                              <xs:enumeration value="Dedo_I_1"/>
                              <xs:enumeration value="Dedo_I_2"/>
                              <xs:enumeration value="Dedo_I_3"/>
                              <xs:enumeration value="Dedo_I_4"/>
                              <xs:enumeration value="Dedo_I_5"/>
                            </xs:restriction>
                          </xs:simpleType>
                        </xs:element>
                        <xs:element name="Involucrado" type="xs:boolean"/>
                        <xs:element name="Posicion">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="X" type="xs:double"/>
                              <xs:element name="Y" type="xs:double"/>
                              <xs:element name="Z" type="xs:double"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

### A.2.2 Vista grafica del XML.

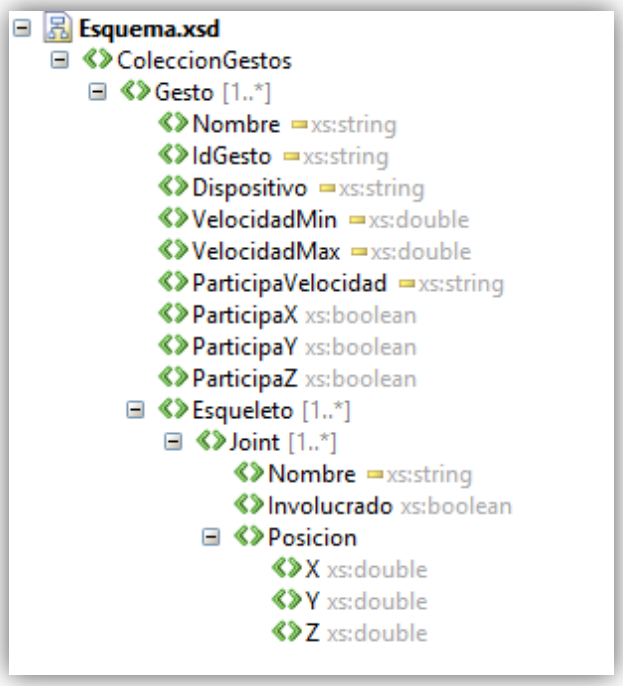



**Figura 83: Vista grafica del formato para el archivo XML.**

# A.3. El visualizador.

## A.3.1. Código fuente.

Se puede obtener el código abierto en el siguiente enlace:

https://codepen.io/leapmotion/pen/yjndm

## A.3.2. Compatibilidad.

Uno de los problemas del visualizador es que depende del navegador, y por lo tanto si no es compatible con el navegador por defecto del usuario, no se podrá visualizar la o las manos.

| **Navegador** | **Compatibilidad** |
|---|---|
| **Mozilla Firefox** | SI |
| **Google Chrome** | SI |
| **Internet Explorer** | NO |

Este documento esta firmado por

| | | |
|---|---|---|
|  | **Firmante** | CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES |
| | **Fecha/Hora** | Fri Jun 06 21:41:16 CEST 2014 |
| | **Emisor del Certificado** | EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES |
| | **Numero de Serie** | 630 |
| | **Metodo** | urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature) |